Semantic Operations for Transfer-based Machine Translation

Michael Wayne Goodman

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2018

Reading Committee:

Emily M. Bender, Chair

Francis Bond, Chair

Fei Xia

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Semantic Operations for Transfer-based Machine Translation

Michael Wayne Goodman

Co-Chairs of the Supervisory Committee:
Chair Emily M. Bender
Department of Linguistics

Chair Francis Bond
Division of Linguistics and Multilingual Studies,
Nanyang Technological University

This dissertation describes a new approach to the automatic extraction of semantic mappings (**transfer rules**) for rule-based machine translation. This approach continues previous work in combining HPSG rule-based grammars, whose precise bidirectional implementation facilitates deep semantic analysis of sentences and the enumeration of grammatical realizations of semantic representations, and data-driven techniques of machine translation, whose automatic extraction of knowledge and statistical inference allow models to be quickly built from bitexts and to rank extracted patterns by their frequency. I define two new methods for bilingually aligning semantic fragments (or **semantic subgraphs**) and a heuristic strategy for aligning nodes between source and target subgraphs, which together allow me to design transfer systems that meet, and at times exceed, the translation coverage and quality of the prior state of the art with a significantly reduced dependence on idiosyncratic language-pair definitions (i.e., improved language independence). These improvements are made possible by a number of **semantic operations**, either designed or implemented by me and defined within this dissertation, that fully model the semantic representations and allow for inspection and transformation as graph operations. I apply my methods to the task of translating Japanese sentences into English—a typologically distant language pair.

TABLE OF CONTENTS

Page

List of Figures v

List of Tables viii

Glossary xi

Notational Conventions xviii

Chapter 1: Introduction 1
1.1 Translating Human Language 3
1.2 Machine Translation using Semantic Representations 6
1.3 Main Contributions 10
1.4 Document Outline 12

Chapter 2: DELPH-IN Semantics 15
2.1 Philosophical Basics 17
2.2 Structural Semantic Preliminaries and Terminology 19
2.3 Underspecified Representation 22
2.4 Visual Presentation of DELPH-IN Semantics 24
2.5 SEM-I: The Semantic Interface 25
2.6 MRS: Minimal Recursion Semantics 28
2.7 RMRS: Robust Minimal Recursion Semantics 31
2.8 EDS: Elementary Dependency Structures 31
2.9 DM: Bilexical Dependencies 33
2.10 DMRS: Dependency Minimal Recursion Semantics 34
2.11 Summary of DELPH-IN Representations 36
2.12 Comparison to Other Frameworks 36

2.13 Chapter Summary 39

Chapter 3: Machine Translation 41
3.1 Rule-based Machine Translation 41
3.2 Statistical and Neural Machine Translation 42
3.3 Chapter Summary 45

Chapter 4: System Overview 46
4.1 Translation Pipeline 46
4.2 Pipeline Information Management 49
4.3 Parsing 50
4.4 Transfer 51
4.5 Generation 56
4.6 Translation Selection 56
4.7 Chapter Summary 58

Chapter 5: Semantic Operations 59
5.1 MRS to DMRS Conversion 59
5.2 DMRS to MRS Conversion 64
5.3 Semantic Tests 68
5.4 Semantic Graph Traversal 76
5.5 PENMAN Serialization 82
5.6 Subgraph Extraction 86
5.7 DMRS Simplification 90
5.8 Chapter Summary 95

Chapter 6: Bilingual Semantic Subgraph Alignment 96
6.1 Bilingually Aligned Predicate Phrases 97
6.2 Top-Down Subgraph Enumeration 102
6.3 Filtering Subgraphs and Subgraph Pairs 111
6.4 Related Work 112
6.5 Chapter Summary 115

Chapter 7: Transfer Grammar Augmentation 116
7.1 The LOGON Transfer Machinery 117
7.2 Challenges and Strategies in Building Transfer Grammars 121
7.3 Converting Subgraphs to Transfer Rules 124
7.4 Transfer Grammar Maintenance 131
7.5 Chapter Summary 132

Chapter 8: Data Exploration 133
8.1 Data Sources and Divisions 133
8.2 Basic Analysis 138
8.3 Analysis of Parsing Performance 145
8.4 Analysis of Generation Performance 151
8.5 The Bisem Corpus 152
8.6 Chapter Summary 154

Chapter 9: Experimental Design 155
9.1 Evaluation Metrics 155
9.2 Pipeline Parameters for Transfer-based Systems 158
9.3 Data Preparation 159
9.4 Moses Baseline 160
9.5 Haugereid and Bond Baseline 162
9.6 LPA: Bilingually-aligned Predicate Phrases 164
9.7 SGA: High-frequency Coincident Subgraphs 170
9.8 Chapter Summary 179

Chapter 10: Results 180
10.1 Baseline Development Results 181
10.2 LPA Development Results 183
10.3 SGA Development Results 185
10.4 Combined Test Results 185
10.5 Translation Examples 189
10.6 Chapter Summary 191

Chapter 11: Analysis 192
11.1 Translation Analysis 192
11.2 Comparing Systems by Automatic Quality Estimations 195
11.3 SGA Subgraph Topologies 201
11.4 Semantic Analysis 204
11.5 Performance Comparison 206
11.6 Translation Examples 211
11.7 Conclusion 215

Chapter 12: Conclusion 218
12.1 Methodological and Artifactual Contributions 219
12.2 Next Steps and Future Research 222
12.3 Closing Words 228

Bibliography 229

Appendix A: Setting Up the Translation Environment 245
A.1 Moses 245
A.2 ERG 247

Appendix B: Database Schema 249

LIST OF FIGURES

Figure Number — Page

1.1 Vauquois triangle . . . 7
1.2 Vauquois inverted funnel with a very long spout . . . 8

2.1 MRS graph view for *The dog sleeps.* . . . 16
2.2 MRS graph view for *Every dog chased some cat.* . . . 25
2.3 Variable hierarchy . . . 26
2.4 MRS fragments of *the large and gentle dog sleeps* . . . 32
2.5 DMRS fragments of *the large and gentle dog sleeps* . . . 35
2.6 Ordering of information density among MRS, DMRS, EDS, and DM . . . 37

4.1 Simplified translation pipeline . . . 46
4.2 Pipeline fan-out . . . 48
4.3 Expanded view of the parsing component . . . 50
4.4 Monolingual MTR in JaEn for normalizing alternate orthographies . . . 53
4.5 Hand-written MTR for the idiomatic 嘘をつく *uso-wo tsuku* "tell a lie [lit: breathe a lie]" . . . 53
4.6 Building the transfer pair store . . . 54
4.7 Expanded view of the transfer component . . . 55
4.8 Expanded view of the generation component . . . 56
4.9 Expanded view of the selection component: First method . . . 57
4.10 Expanded view of the selection component: Oracle method . . . 57
4.11 First-selection of translation hypotheses . . . 57

5.1 MRS for *The dog whose tail is long barked.* . . . 62
5.2 DMRS for *The dog whose tail is long barked.* . . . 63
5.3 Alternative DMRS for *The dog whose tail is long barked.* . . . 63
5.4 Before and after the string transformation for structural isomorphism checking 71
5.5 Multiply-rooted DMRS for *The southbound train departed.* . . . 72

5.6 DMRS subgraphs showing non-scopal modification 75
5.7 DMRS subgraphs comparing (a) scopal and (b) non-scopal modification . . . 75
5.8 Multiply-rooted DMRS for *The angry farmer fanatically chased the rabbit* . 80
5.9 Rooted DMRS traversal order with inverted links 80
5.10 DMRS triples for *the dog whose tail is long barked* 84
5.11 PENMAN serialization for *the dog whose tail is long barked* 85
5.12 ID-normalized PENMAN serialization for *the dog whose tail is long barked* . 85
5.13 DMRS graph for *Pierre Vinken, 61 years old* 91
5.14 Removing implicit quantifiers 92
5.15 Simplifying predications with constant values 93
5.16 Converting binary predications to links 94
5.17 Final simplified DMRS graph 94

6.1 Original and simplified DMRS graphs for エチオピア中部に位置する 98
6.2 Original and simplified DMRS graphs for *located in central Ethiopia* 99
6.3 Example alignment for エチオピア中部 → central Ethiopia 102
6.4 Example subgraph pair extracted using the alignment in Fig. 6.3 102
6.5 Example Japanese subgraphs enumerated for Fig. 6.1a 104
6.6 Example English subgraphs enumerated for Fig. 6.2a 105
6.7 Example prefilters 106
6.8 Example ϕ^2 values 111

7.1 TDL definition of `mrs_transfer_rule` in JaEn 118
7.2 TDL definition of `monotonic_mtr` and `monotonic_omtr` in JaEn 119
7.3 TDL definition of `optional_mtr` in JaEn 119
7.4 Bilingual variable binding for 恐ろしい夢を見る *osoroshii yume-wo miru* "have a terrible dream" 126
7.5 Bilingual variable binding for すぐ医者に電話する *sugu isha-ni denwa suru* "call the doctor right away" 128
7.6 Subgraphs and MTR for 太陽 *taiyou* → *sun* 130
7.7 Subgraphs and MTR for 面白い手紙 *omoshiroi tegami* → *interesting letter* . 130
7.8 Subgraphs and MTR for 濁水 *dakusui* → *murky water* 131

8.1 Number of sentences per token-count. 140
8.2 Parsing coverage for each sentence length. 148

8.3 Average per-item parsing time for each sentence length. 149
8.4 Average per-item parsing memory usage each sentence length. 150

9.1 Correct subgraph pairing for 海王星 *Kaiousei* "Neptune" 171
9.2 Incorrect subgraph pairings for 海王星 *Kaiousei* "Neptune" 171

10.1 Coverage, All-BLEU, and Intersective-BLEU results for all LPA configurations 183
10.2 Coverage, All-BLEU, and Intersective-BLEU results for all SGA configurations 186
10.3 Oracle Translations for the top configurations of LPA 189
10.4 Oracle Translations for the top configurations of SGA 190
10.5 Translations from LPA-O2 and SGA-O5 190

11.1 LPA and SGA overlap with H&B and the most common translations 193
11.2 Cross overlap between LPA and SGA translations 195
11.3 Intersective-First and Intersective-Oracle scores for BLEU and METEOR over all LPA and SGA configurations 197
11.4 Errors in transferred MRSs 205
11.5 Transfer ambiguity in LPA and SGA 207
11.6 Percentage of transfer inputs that hit the timeout or memory limit 208
11.7 Realization ambiguity and timeouts 209
11.8 Partial transfers and lexical gaps 210
11.9 Experiential versus progressive errors in translation 212
11.10 Wrong sense of する *suru* in translation 213
11.11 Literal versus idiomatic errors in translation 213
11.12 Collocation mismatch error in translation 214
11.13 DMRS for 彼は自分の過失の責任を認めた 215
11.14 215

B.1 `item` Table 249
B.2 `p-info` Table 249
B.3 `p-result` Table 250
B.4 `x-info` Table 250
B.5 `x-result` Table 250
B.6 `g-info` Table 251
B.7 `g-result` Table 251

LIST OF TABLES

Table Number Page

2.1 Feature matrix of the prominent *MRS variants . . . 37

5.1 Nodes and surface pointers for *machine translation* . . . 77
5.2 Nodes and surface pointers for *isn't* in *Kim isn't a student* and *unwound* in *Kim unwound the string* . . . 78
5.3 Default and obligatory quantifiers in the ERG and Jacy . . . 92

6.1 Basic statistics of enumerated subgraphs . . . 108
6.2 Bilingual contingency table . . . 109

8.1 Data splits for the Tanaka Corpus . . . 135
8.2 Data splits for the Kyoto Corpus . . . 136
8.3 Data splits for the Japanese WordNet Corpus . . . 137
8.4 Data splits for all corpora . . . 138
8.5 Sentence and token counts for the corpora . . . 139
8.6 Number of sentences with a maximum of 70, 35, or 20 tokens . . . 141
8.7 Duplicate sentences in training data . . . 143
8.8 Duplicate sentences in development data . . . 143
8.9 Duplicate sentences in test data . . . 144
8.10 Duplicates of testing data in the training and development data . . . 145
8.11 Parsing constraints for ACE . . . 146
8.12 Machine information . . . 146
8.13 Generation performance on the development data . . . 153
8.14 Bisem counts . . . 153

9.1 ACE parameters for parsing, transfer, and generation . . . 159
9.2 Basic settings for the moses baseline . . . 162
9.3 Extracted H&B rule counts by specified predicate counts . . . 164
9.4 Dropped predicates for linearization . . . 165

9.5 Extracted LPA rule counts by subgraph order . . . 167
9.6 Extracted LPA isomorphic rule counts by subgraph order . . . 168
9.7 Experimental configurations for LPA . . . 169
9.8 Dropped predicates for enumeration . . . 172
9.9 Subgraph prefilters . . . 173
9.10 SGA subgraph pair counts by subgraph order . . . 175
9.12 Extracted SGA isomorphic subgraph pair counts . . . 177
9.13 Experimental configurations for SGA . . . 178

10.1 H&B transfer and generation coverage for the development set . . . 182
10.2 Moses evaluation results over the development data . . . 182
10.3 H&B evaluation results over the development data . . . 182
10.4 Coverage and All-BLEU for top LPA configurations . . . 184
10.5 Top LPA configurations' intersective evaluation results . . . 185
10.6 Coverage and All-BLEU for top SGA configurations . . . 186
10.7 Top SGA configurations' intersective evaluation results . . . 187
10.8 Coverage and All-BLEU for testing configurations . . . 188
10.9 Evaluation results over the intersection of test translations . . . 189

11.1 System-level overlap with Moses and the reference translation . . . 196
11.2 Difference in normalized BLEU, NIST, and METEOR between Moses and my systems . . . 200
11.3 Graph orders and numbers of structural variants . . . 202
11.4 Percentage of subgraphs with the given structure . . . 203
11.5 Most frequent order-3 structures, with roles, in Jacy and the ERG . . . 203

LIST OF ALGORITHMS

1 Converting MRS EPs to DMRS Nodes 61

2 Finding the indices of representative EPs 61

3 Converting MRS EPs and qeqs to DMRS Links 65

4 Converting from DMRS to MRS 67

5 Determine if a DMRS link is *to*-oriented 73

6 Converting DMRS nodes and links to triples 83

7 Extracting DMRS subgraphs from a predicate list 87

8 Extracting DMRS subgraphs from a restricted graph traversal 89

9 Extracting DMRS subgraphs using predicate alignments 101

10 Pairing DMRS subgraphs 106

GLOSSARY

ABSTRACT REPRESENTATION: a linguistic representation that uses coarse-grained and generalized category labels; cf. **concrete representation**

ABSTRACT PREDICATE: a **semantic predicate** that does not correspond directly to a surface token (e.g., `compound` for forming compound phrases, or `neg` for generalizing over a variety of negation strategies); cf. **surface predicate**

ARBORESCENCE: a directed, rooted tree

ARGUMENT LINKS: DMRS links corresponding to MRS role arguments; cf. **non-argument links**

BILINGUALLY CORRELATED: when a source and target representation are empirically likely to be translationally equivalent

BILINGUALLY ISOMORPHIC: when a source and target representation are structurally isomorphic

BILINGUAL VARIABLE BINDING: the coindexation of variables in source and target representations

BOUND VARIABLE: a variable bound by a quantifier

BISEM: a bilingual semantic corpus

CHARACTERISTIC VARIABLE: see **intrinsic variable**

CHARACTERIZATION: see **surface alignment**

COMPOSITIONAL REPRESENTATION: a representation that is monotonically built up from smaller pieces that are each licensed by an input token; cf. **non-compositional representation**

CONCRETE REPRESENTATION: a representation that is fully specified for an application or domain-specific usage; cf. **abstract representation**

CONNECTEDNESS: the graph property of having an undirected path between any two nodes in the graph

DELPH-IN SEMANTICS: the semantic framework used by DELPH-IN grammars and resources, most notably Minimal Recursion Semantics (MRS; Copestake et al., 2005)

DEPENDENT: in semantic dependency representations, the side of a relation that corresponds to a functor's argument; cf. **head**

DISTINGUISHED VARIABLE: see **intrinsic variable**

ELEMENTARY PREDICATION (EP): an MRS construct that combines a **semantic predicate** with its associated arguments

EP CONJUNCTION: a set of EPs that share a quantifier scope

FIRST SELECTION: a method of selecting one **translation hypothesis** among many by taking the first realization; cf. **Oracle selection**

FROM-ORIENTED LINK: a DMRS link connecting a quantifier to its quantifiee or a non-scopal modifier to its modifiee; a **rooted traversal** would start at the link's *to* node and end on the *from* node; cf. ***to*-oriented link**

GRAPH ORDER: the number of nodes in a graph

HANDLE CONSTRAINT: specified relationships between **holes** and **scope handles**; most notably, **qeq**

HEAD: in semantic dependency representations, the side of a relation that corresponds to a functor; cf. **dependent**

HOLE: an argument position in underspecified representations that is understood to be filled with a **scope handle**

HYPOTHESIS: a selection of parse result, transfer result, and generation result that results in a translation

HYPOTHESIS SET: the set of **hypotheses** for an input item

IDENTICAL: a duplicate item in a corpus that is part of the natural distribution; cf. **spurious duplicate**

INDIVIDUAL CONSTRAINT: specified relationships between the variables for individuals; used, for instance, to encode information structure

INTERLINGUA: a (usually high-level, such as semantic) representation that is shared between two or more languages

INTERSECTIVE SUBSET: in a comparison of translation systems with imperfect cover, the subset of results that are covered by all systems

INTERSET DUPLICATE: a duplicate that repeats an item appearing in another dataset

INTRASET DUPLICATE: a duplicate that repeats an item within the same dataset

INTRINSIC VARIABLE: the variable (canonically with the ARG0 role) in an **elementary predication** that represents the predication itself; for individuals it is the variable bound by quantifiers, and for all predications it can be used to select an argument

INTRINSIC VARIABLE PROPERTY: a requirement that every **elementary predication** has a unique **intrinsic variable**

INVERSE LINK: a DMRS link that has been marked to indicate that its preferred direction of traversal in a rooted graph is opposite to its **link direction**

ISOMORPHISM: graph comparison that considers if two graphs have the same shape, often including node and edge labels; also see **structural isomorphism**

LABEL: see **scope handle**

LEXICAL GAP: a error in sentence generation where an input **semantic predicate** corresponds to no known lexical entry in the grammar

LHEQ: a relation between a **hole** and **scope handle** that encodes a strict ordering

LINK DIRECTION: the direction of a link determined by its functor-argument (i.e. *from*→*to*) order

LINK ORIENTATION: a property of DMRS links that indicates the preferred direction in a **rooted traversal** and is separate from the inherent functor-argument direction of the link

LNK VALUE: see **surface alignment**

MEMOUT: a processing condition where a process runs out of allocated memory

MINIMAL RECURSION SEMANTICS (MRS): (Copestake et al., 2005) a semantic representation characterized by its flat structure and underspecified quantifier scope; commonly used in conjunction with HPSG grammars

MONOLINGUAL GRAMMAR: an implemented grammar for parsing/generating sentences for some language (also called a **grammar** if the usage is unambiguous)

MORPHOSEMANTIC PROPERTY: a property attached to a predication, node, or variable that encodes the semantic effects of morphology or directly from lexical items (e.g., tense on events, number on individuals)

MULTI-WORD EXPRESSION RULE: a transfer rule capturing more than one predicate on the source side

NON-ARGUMENT LINK: DMRS links that do not correspond to MRS role arguments; cf. **argument links**

NON-COMPOSITIONAL REPRESENTATION: a representation that is not built-up by individually licensed components; cf. **compositional representation**

NUMERIC REPRESENTATION: a representation that uses real values; cf. **symbolic representation**

OCCASION MEANING: the meaning of a sentence as was originally intended by its speaker

ORACLE SELECTION: a method of selecting one **translation hypothesis** among many by taking the realization that best fits a reference translation, e.g., against a metric such as BLEU; cf. **First selection**

OUTSCOPES: an relation between a **hole** and **scope handle** that is an alternative to **qeq** and **lheq**, although its implementation may vary

PREDICATE: see **semantic predicate**

PREDICATE PHRASE: an n-gram of contiguous semantic predicates

PREDICATE PHRASE PAIR: a pairing of source and target **predicate phrases**

PREDICATION: a **semantic predicate** and its associated arguments

QEQ: equality modulo quantifiers; a relation between a **hole** and **scope handle** that encodes a partial ordering

REPRESENTATIVE NODE: in a scopal conjunction, the predication that is most salient as representing the entire set

ROLE: see **semantic role**

ROOTED TRAVERSAL: a traversal of a DMRS graph that results in a singly-rooted graph

SCOPAL MODIFICATION: modification where the argument is a **scope handle** rather than a particular predication

SCOPE HANDLE: a label that refers to a scope position or scopal conjunction

SEMANTIC GRAPH: a semantic representation viewed as a graph of predicates and their arguments

SEMANTIC INTERFACE: a description that relates a semantic model to a syntactic model

SEMANTIC MODEL: a system of semantic predicates, roles, properties, and variables, as well as constraints on well-formed structures

SEMANTIC PREDICATE: a symbol in a semantic representation that signifies an entity or construction

SEMANTIC OPERATIONS: procedures that inspect or transform semantic representations

SEMANTIC ROLE: the semantic relationship between a functor (e.g., **semantic predicate**) and its argument

SEMANTIC SPACE: the range of reference, or the intension, for a particular word sense

SEMANTIC SUBGRAPHS: a fragment of a larger semantic representation

SEMANTIC TRANSFER: **transfer** where the material being shifted is a semantic representation

SINGLE RULE: a **transfer rule** that matches one source predicate

SPEAKER MEANING: see **occasion meaning**

SPURIOUS DUPLICATE: a duplicate item in a corpus that is not part of the natural distribution—e.g., caused by user error; cf. **identical**

STANDING MEANING: the meaning of a sentence as can be inferred by its surface form without contextual information

STRUCTURAL REPRESENTATION: a representation that links information together into a larger structure

STRUCTURAL ISOMORPHISM: graph comparison that only considers nodes, edges, and edge labels, not node labels

SURFACE ALIGNMENT: a correspondence between semantic material and positions in the linear linguistic signal; in *MRS representations this is given by the CFROM and CTO values (also called **characterization** or **lnk values**)

SURFACE PREDICATE: a **semantic predicate** corresponding to a token in the surface form

SYMBOLIC REPRESENTATION: a representation using discrete, atomic symbols that stand in for unknown real values (cf. **numeric representation**)

TO-ORIENTED LINK: a DMRS link such that a **rooted traversal** would start on its *from* node and end on its *to* node; cf. ***from*-oriented link**

TRANSFER: the step of **translation** where material shifts from a source-language representation to a target-language representation (see also **semantic transfer**)

TRANSFER GRAMMAR: a system of rules for mapping one semantic representation to another

TRANSFER GRAMMAR AUGMENTATION: the process of building upon a small, core transfer grammar with additional transfer rules

TRANSFER PAIR STORE: a set of accumulated transfer pairs

TRANSFER RULE: a rewrite rule that maps source language material to target language material

TRANSLATION: the process of producing one or more target-language sentences for a given source-language sentence

UNDERSPECIFIED REPRESENTATION: a representation that accommodates ambiguity by generalizing over fully specified forms

VARIABLE PROPERTY: a **morphosemantic property** that is associated with an MRS variable

VARIABLE-PROPERTY MAPPING: a system of rules that transforms the grammar-internal **variable properties** and values to and from grammar-external properties and values

NOTATIONAL CONVENTIONS

Typesetting

In regular prose, I use the typesetting conventions in the following table:

Convention	Usage	Example
Italics	cited linguistic forms,[a]	…allows *61 years* to modify…
	transliteration	太陽 *taiyou* → *sun*
"Quotes"	in-text glossing	エチオピア *Echiopia* "Ethiopia"
	direct quotes	…had asked, "what is the…
Slanted	math, algorithms, variables	…each alignment $a \in A$ is…
Bold	emphasis,	31.5 24.6 **43.0** 10.6
	introduced terminology	…bilingual semantic (**bisem**) corpus…
SMALL-CAPS	grammatical glosses,	…Touzai -line -GEN opening -DAT…
	feature names,	…features include FLAGS.EQUAL and…
	functions	…function DMRSNODEDEPTHS() takes…
`Monospace`	semantic predicates,	…use of `number_q` would…
	code, serialized data	`(e0 / _find_v_1)`

[a] forms written in Roman script; forms in non-Roman orthographies use upright fonts

There are additionally some project names that are typeset specially, such as LOGON, Verb*mobil*, [incr tsdb()], and Zhong `[|]`. In algorithms, I use **bold** for control-flow syntax and keywords, *slanted* for variables and data structures, SMALL CAPS for defined functions, `monospace` for string literals, upright serif for basic operations, and right-alignment for comments. For example:

if ToOrientedLink(l) **then**

$relation \leftarrow$ FormatString("%s-%s",l^{role},l^{post}) ▷ E.g., `ARG1-NEQ`

append $\langle l^{from}, relation, l^{to} \rangle$ to T

Glossing

Many linguistic examples are presented as interlinear glossed text (IGT). For example:

> その 単語 を 辞書 で ひい て ごらん。
> *sono tango -wo jiten -de hii -te -goran*
> that word -ACC dictionary -INS look.up -INF -try
> "Look up the word in your dictionary." [jpn]

The first line is a Japanese sentence in native orthography, followed by a transliteration (i.e., romanization) that shows the morphologial boundaries, then the morpheme glosses, and finally a free translation. The transliteration system I use is Kunrei-shiki with the allowed exceptions for, e.g., つ *tsu*, じょ *jo*, を *wo*, etc.[1] I follow the Leipzig Glossing Rules[2] and supplement it with additional glossing grams where necessary, such as NUMCL for numeral classifiers and HON for honorifics.[3] I mark verbal *te*-forms as infinitives (-INF) and I treat case-marking particles as affixes. I use English glosses where the English word is adequate, as in その *sono* "that", but for pronouns I use the more general person-number-gender values (e.g., 君 *kimi* is glossed as 2SG), as English does not always have good overlap with Japanese where values are underspecified.

The table on the following page lists glosses of grammatical categories (i.e., gloss grams). Note that I only include those that are used in the document.

---

[1] See `http://www.mext.go.jp/b_menu/hakusho/nc/k19541209001/k19541209001.html` (Japanese).

[2] `https://www.eva.mpg.de/lingua/resources/glossing-rules.php`

[3] The honorifics system in Japanese is much more nuanced than I present in my examples. For example, Siegel et al. (2016) use six different grams for nominal affixes, addressee honorifics (positive and negative), and subject honorifics (positive, neutral, and negative), but I ignore these complexities as the distinctions are not directly relevant for discussion in this document.

Gram	Meaning	Example
2	second person	君 *kimi* “you”
3	third person	彼 *kare* “he/him”
ACC	accusative	を *wo*
ADN	adnominal	の *no*
ADV	adverbializer	に *ni*
COMP	complementizer	と *to*
COP	copula	だ *da*
DAT	dative	に *ni*
DEO	deontic	べき *beki*
GEN	genitive	の *no*
HON	honorific	です *desu*
INF	infinitival	て *te*
INS	instrumental	で *de*
LOC	locative	に *ni*
M	masculine	彼 *kare* “he/him”
NMLZ	nominalizer	の *no*
NOM	nominative	が *ga*
NUMCL	numeral classifier	人 *nin*
ORD	ordinal	番目 *banme*
PFV	perfective	た *ta*
Q	interrogative	か *ka*
REFL	reflexive	自分 *jibun*
SG	singular	君 *kimi* “you”
SUPL	superlative	もっとも *mottomo* “most”
TOP	topic	は *wa*

Sets and Data

This document often describes operations in terms of sets and with data structures as tuples with named indices. I use **set-builder notation** for describing sets and a similar parenthesis-delimited notation for describing lists (i.e., a **list comprehension**). The following table covers the data structures used to describe my algorithms:

Form	Usage	Example
$\langle \ldots \rangle$	tuple	a $\langle$CFROM, CTO$\rangle$ (i.e., start and end) pair…
$\{\ldots\}$	set	**if** $l^{post} \in \{$`"NEQ"`, `"EQ"`$\}$ **then**
$(\ldots)$	list	…if M is $(\{a, b\}, \{c\}, \{a, b\})$, there will be…

The following table describes the notation I use to refer to data via variables:

Form	Usage	Example
x	atomic values, tuple instances	l is a Link as a $\langle from, to, role, post \rangle$ tuple
x^y	labeled element in tuple	**if** n^{carg} is not nil **then**
X	sets, lists	…a list of EPs R and a…
X_i	list indexing	Upon visiting a node N_i, the…
X^y	labeled elements in list of tuples	$(l^{from} \in N'^{id})$ and $(l^{to} \in N'^{id})$
X_i^y	the y element from the ith tuple in X	$from \leftarrow N_i^{id}$

Note that variable names can be more than one letter, but atomic values and tuples will be lowercased (e.g., *role*), while sets and lists will start with an uppercase letter (e.g., $Conj$). I also use sets of $\langle$*key*, *value*$\rangle$ tuples as property maps with a function-like syntax for access. For example, if *Arg* is a property map of roles to arguments, *Arg*(`"carg"`) $\leftarrow N_i^{carg}$ sets the value of the key `carg` in the mapping *Arg* to the value at N_i^{carg}.

ACKNOWLEDGMENTS

I would first like to thank my committee. First I thank my advisor, Emily M. Bender, for her tireless help with this project as well as the many other projects I worked on as her student. I often relied on her vast knowledge of our field and impressive ability to quickly review papers (sometimes while walking to or from campus) for things as small as punctuation and as large as document structure and framing. She also had insightful suggestions for improving methodologies, evaluations, and relations to previous work; a strong sense of personal and professional ethics; and a fearless and unapologetic determination to make our societies—academic as well as local, national, and global—better for all involved. I thank my co-advisor, Francis Bond, who not only provided a wealth of experience in East and Southeast Asian languages, lexical resources, machine translation, and more, but he, along with Kyonghee and Arthur, have been dear friends to my family for many years. I thank Fei Xia for her insight into other subfields and for the many research opportunities she offered me during my studentship. I also thank Luke Zettlemoyer for his questions during and (along with all above) flexibility in scheduling my defense.

In addition, I'd like to thank: Ann Copestake and Guy Emerson for answering my endless questions about MRS and DMRS; Stephan Oepen for his collaboration on various projects and insightful and detailed replies to my questions; Dan Flickinger for his help with the ERG and for *nevertheless*; František Kratochvíl for his help with language resources and with navigating NTU's administration; Luis Morgado da Costa for his suggestions and opinions (except, maybe, regarding dessert); Toby,

Rafe, Arthur, and Matt for evaluating my system outputs; Matic Horvat for first rekindling the dormant JaEn back to life and offering useful suggestions for going forward; Woodley Packard for providing support for ACE, discussions, an RAship, and for hosting informal and useful mini-research-summits; Glenn Slayden for useful discussions and also for hosting a mini-summit; Michael Jellinghaus for providing a copy of his hard-to-find dissertation and for kicking off work on the automatic extraction of transfer rules from bilingual corpora; Eric Nichols for the initial work of augmenting JaEn with automatically created rules and for creating the transfer setup that inspired my own pipeline; Petter Haugereid for building the system that I used as a baseline and for providing insight when building my own system; Ioannis Konstas for useful discussions and experiments with neural generation from DMRS; Francis Bond, Guy Emerson, Alex Kuhnle, and T.J. Trimble for their contributions to PyDelphin; Daniel Hershcovich and Marcos Pertierra for their contributions to the Penman library; David Brodbeck and Brandon Graves for IT support; Mike Furr, Joyce Parvi, and Cathy Carrera for all their administrative help (and occasional commiserating); Rik Koncel-Kedziorski and Marina Oganyan for their suggestions after my practice-defense; and the members of DELPH-IN, members of BondLab and the Linguistics and Multilingual Studies group at Nanyang Technological University, as well as the members of emb-students and the Linguistics Department at the University of Washington, for their all support.

Finally, I wish to thank my family: my parents and parents-in-law for all their encouragement and financial support; my sons, for always being happy to see me when I got home; and my wife, Ning, for her love, patience, and good humor throughout this project.

DEDICATION

to my wife, Ning, and to my sons, ⟨Willamette, {Rainier, Emmett}⟩

Chapter 1

INTRODUCTION

This dissertation describes a new approach to the automatic extraction of semantic mappings for rule-based machine translation. This approach continues previous work in combining HPSG (Pollard and Sag, 1994) rule-based grammars, whose precise bidirectional implementation facilitates deep semantic analysis of sentences and the enumeration of grammatical realizations of semantic representations, and data-driven techniques of machine translation, whose automatic extraction of knowledge and statistical inference allow models to be quickly built from bitexts and to rank extracted patterns by their frequency. The grammars map strings to an underspecified **semantic model**, viz., Minimal Recusion Semantics (MRS; Copestake et al., 2005), but this model is not an **interlingua**, or shared representation, between the source and target grammar; rather, each grammar defines its own model. Having per-grammar semantic models allows the grammars to capture the semantic realities of the target language without having to make concessions or compromises in order to accommodate another, possibly very different language, but it also means that semantics-based translation must work with two different models. I therefore work in the space of **semantic transfer** where **transfer rules** map semantic fragments in the source grammar's semantic model to fragments in the target model. By focusing only on the transfer of semantics, i.e., the translation of meaning, I can rely on the source and target grammars to perform the mapping to and from semantic representations.

Ideally, a **transfer grammar** (a system of transfer rules and supporting definitions) would map the entirety of the source semantic model to the target model, but hand-building such a grammar—as the monolingual grammars are hand-built—is impractical. This impracticality is because (1) it is only useful for one language pair and in one direction (unlike

monolingual grammars, which can be useful as the source or target of a translation system); (2) it needs to be updated whenever either the source or target grammar changes a semantic analysis; (3) its author needs to be highly skilled: proficient in both languages and familiar with both implemented grammars; and (4) in order to have reasonable transfer coverage it needs a large number of rules, and the effort to create such a system likely exceeds the capabilities and/or interest of those with the requisite skills. In contrast, a data-driven transfer-rule extractor can be used to generate a large number of rules quickly and it therefore becomes trivial to update a transfer grammar to changes in the monolingual grammars (or to, say, create a domain-specific transfer grammar by selecting subsets of the training data) and it does not require a highly-skilled grammar writer. My approach is **transfer grammar augmentation**—automatically extracting a large number of transfer rules and using them to build on top of a small hand-built **core transfer grammar** which defines the basic structures that the extracted rules make use of.

I define two new methods for bilingually aligning semantic fragments (or **semantic subgraphs**) and a heuristic strategy for aligning nodes between source and target subgraphs, which together allow me to design transfer systems that meet, and at times exceed, the translation coverage and quality of the prior state of the art with a significantly reduced dependence on idiosyncratic language-pair definitions (i.e., improved language independence). These improvements are made possible by a number of **semantic operations**, either designed or implemented by me and defined within this dissertation, that consider the graphical nature of the semantic representations and allow for inspection and transformation as graph operations.

I apply my methods to the task of translating Japanese sentences into English. This language pair is a good choice partly because Japanese and English are typologically distant, so translation between them is more challenging than for some other pairs. This pair is also a convenient choice as two of the largest implemented HPSG grammars already exist for them, and furthermore I'm a native speaker of English and a reasonably competent speaker of Japanese, so I can judge the quality of the extracted rules and the outputs during

development.

For the remainder of this chapter, I will discuss challenges in translating human language in Section 1.1, motivate my choice of machine translation paradigm in Section 1.2, and list the main contributions of the thesis in Section 1.3. I close the chapter in Section 1.4 with an outline of the remaining chapters.

1.1 Translating Human Language

The translation of human languages is an incredibly complex task. A translator endeavors to understand the meaning and intent of the source utterance then produce a new utterance in the target language whose content has the same meaning—and intent—as the original. In addition, the translation ideally matches the original tone (e.g., emotion, politeness), and is adapted for the linguistic competence of the audience or interlocutor (who might not be familiar with certain jargon, or might not even be native speakers of the target language). For spoken translation, this also needs to occur within a very short span of time.

Languages are embedded in a culture, or perhaps across many cultures, and this affects a speaker's use of idiom, metaphor, allegory, etc. Furthermore, words in one language do not always have a simple, single-word translation in another. In order to get the appropriate meaning across, a source word may therefore be translated as multiple words, or perhaps not translated at all. Even within a single language, a word often has multiple senses. For instance, the English word *light* has many nominal, verbal, and adjectival senses: the thing that illuminates (i.e., photons; *this room gets lots of light in the morning*), a kind of electromagnetic radiation, a device that produces light, the act of igniting something (e.g., cigarettes; *can I get a light?*), to illuminate, to ignite, bright, pale, etc. In addition, there are senses having to do with weight or significance (e.g., *a light bag*, *a light meal*, *a light touch*, *a light jacket*, *a light suggestion*) or a verb meaning *to descend*, *to get down*, etc. While these senses all share the same form in English, each one may result in a different form in another language, although often some senses correspond to the same form cross-lingually because they are derivative. For instance, 心 *kokoro* "heart" in Japanese means the cardiovascular

organ, but also has the senses *mind* or *core*, as in English. Even for a single word sense there is a semantic space or range of reference it occupies, figuratively, in a speakers mind. For example, *water* in English is the liquid form of H_2O, whereas in Japanese the word 水 *mizu* is generally used only for cold, cool, or tepid water, while 湯 *yu* is used for hot water. The imperfect overlap of words across languages is one challenge for translation, although the problem is made easier by considering more than one word at a time as word collocations which can resolve some lexical ambiguities.

Beyond lexical translations, there is also the problem of word-order differences. Some differences are localized within a clause, such as the adjective-noun versus noun-adjective difference in English and Spanish (e.g., *red tomato* versus *tomate rojo*), but sometimes more is involved than just swapping a pair of words. For example, Japanese is generally similar to English in putting adjectives before nouns, but Japanese puts nearly all nominal modifiers before nouns, including clausal modifiers, where English does not.[1] These differences are matters of surface realization; there are no significant differences in the semantics. These kinds of variation can become even more complex, but a full enumeration of typological variation with respect to word order is beyond the scope of the discussion here. HPSG grammars are good at mapping semantic representations to grammatical surface realizations so I can rely on them for these kinds of differences.

More difficult are differences that fundamentally change the structure in some way that goes beyond word sense and word ordering choices. Dorr (1994, p. 598) enumerates a number of linguistic divergences that change the argument structure in the syntax and semantics, shown in (1)–(7).

---

[1]Consider (i), where 太郎が炊いた *Tarou-ga taita* "Tarou cooked" appears before the noun in Japanese, but after in English, although the adjective 美味しい *oishii* "delicious" appears before the noun in both.

(i) 太郎 が 炊い た 美味しい 飯
Tarou -ga tai -ta oishii meshi
Tarou -NOM cook -PFV delicious meal
"The delicious meal that Tarou cooked"

(1) **Thematic divergence:**
I like Mary ⇔ *Maria me gusta a mi* [eng] ⇔ [spa]
"Mary pleases me"

(2) **Promotional divergence:**
John usually goes home ⇔ *Juan suele ir a casa* [eng] ⇔ [spa]
"John tends to go home"

(3) **Demotional divergence:**
I like eating ⇔ *Ich esse gern* [eng] ⇔ [deu]
"I eat likingly"

(4) **Structural divergence:**
John entered the house ⇔ *Juan entro en la casa* [eng] ⇔ [spa]
"John entered in the house"

(5) **Conflational divergence:**
I stabbed John ⇔ *Yo le di puñaladas a Juan* [eng] ⇔ [spa]
"I gave knife-wounds to John"

(6) **Categorial divergence:**
I am hungry ⇔ *Ich habe Hunger* [eng] ⇔ [deu]
"I have hunger"

(7) **Lexical divergence:**
John broke into the room ⇔ *Juan forzo la entrada al cuarto* [eng] ⇔ [spa]
"John forced (the) entry to the room"

In (1), the Spanish *gustar* (in its conjugated form *gusta*) is translationally equivalent to the English *like*, but the arguments are opposite of those for *like*.[2] In (6), the English *hungry* is a state (i.e., an adjective), but the German *Hunger* is something one has (i.e., a noun). My methodology is well-suited to address divergences exemplified by (4), (5), and (7) as it can extract generalized rules that capture the phenomenon while leaving the arguments of the verbs open. For the other patterns, it can possibly capture subgraphs that exhibit the phenomena, but these would be specialized rules that would not generalize to variations of the sentences (e.g., *I like Sandy* instead of *I like Mary*).

There are many ways that languages can differ, and sometimes these require different

[2]It structurally more similar to the English *please*, but *please* is a less natural translation.

techniques for improving translation results. For differences in word order and morphological realization, I rely on the HPSG grammars to map strings to and from semantic representations. For lexical mapping and structural semantic divergences, transfer using larger semantic subgraphs (i.e., not just a single word or the equivalent semantic entity) will help capture the appropriate lexical sense and argument structure.

1.2 Machine Translation using Semantic Representations

Transfer is often defined as a method of translation that converts high-level source structures (e.g., syntactic or semantic representations) to the equivalent target representation. I think a broader definition is more useful when comparing machine translation methodologies, so I redefine transfer as the process that maps source-language representations to target language representations. **Translation** is therefore the process of converting a source-language sentence into a target-language sentence, including any preprocessing, analysis, transfer, realization, and postprocessing steps. By this definition, even word-based and phrase-based translation methods include a transfer step and translation via an interlingua would perhaps be the only method that does not. My work targets transfer at the semantic level.

Translating via semantics may have benefits over word-based or even syntax-based translation methods, such as for capturing long-distance dependencies or for abstracting or normalizing over differences that are more important for monolingual modeling than for cross-lingual transfer. Machine translation researchers often appeal to this notion by referencing the Vauquois Triangle (Vauquois, 1968), reproduced in Fig. 1.1. At the base corners of the triangle are the source-language sentence (on the left) and the target-language sentence (on the right). Translation is a path across this triangle that transforms the source sentence to the target sentence. The higher up the *Analysis* side of the triangle a path climbs, the higher-level (i.e., more abstract) the source representation becomes and, in theory, the closer the representation becomes with respect to a target representation. Descending the *Generation* side of the triangle corresponds to the realization of sentences from various levels of abstraction. My definition of transfer is thus any lateral traversal from the *Analysis* side to

the *Generation* side of the triangle. Closeness in representation is expected to correlate with the relative ease of transfer. For example, token transfer (e.g., string-to-string) methods not only have to map tokens to translationally equivalent tokens, but they need to model (or approximate) source and target word orders, morphology, whether or not to drop or insert unmapped tokens, etc. Semantic transfer relies on source and/or target resources, such as implemented grammars, to handle most of these tasks monolingually, and transfer maps only the semantic effects of source-language analysis to a target-language representation. Transfer need not always map to an equivalent level: it can map semantics to strings, syntax to strings (also called tree-to-string), strings to syntax (string-to-tree), and so on, depending on the source or target language resources that are available.

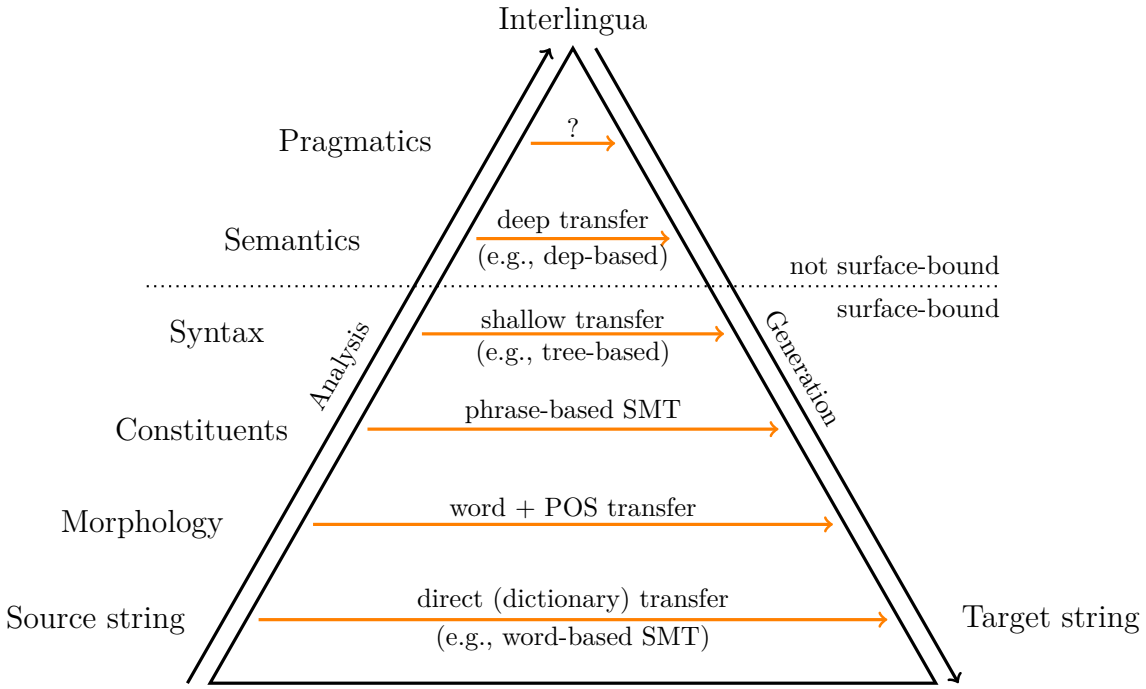


Figure 1.1: Vauquois triangle

One variation on the Vauquois Triangle is a version with no convergence to an interlingua, as shown in Fig. 1.2. This version was created as a reaction to the Vauquois Triangle in order to highlight the difficulty or implausibility of arriving at a common representation.

It is simultaneously attributed to (at least) two sources: Ann Copestake, as the *Copestake Volcano*,[3] although she herself calls it the *Vauquois inverted funnel with a very long spout*;[4] and Satoru Ikehara, as the *Ikehara discontinuity*.[5] Interlinguas are certainly not impossible, especially for controlled domains like technical documentation for heavy machinery (Lonsdale et al., 1994) or weather reports (Chandioux, 1976), but it may be impossible, or at least more effort than it's worth, to create an accurate, expressive, general-purpose interlingua.

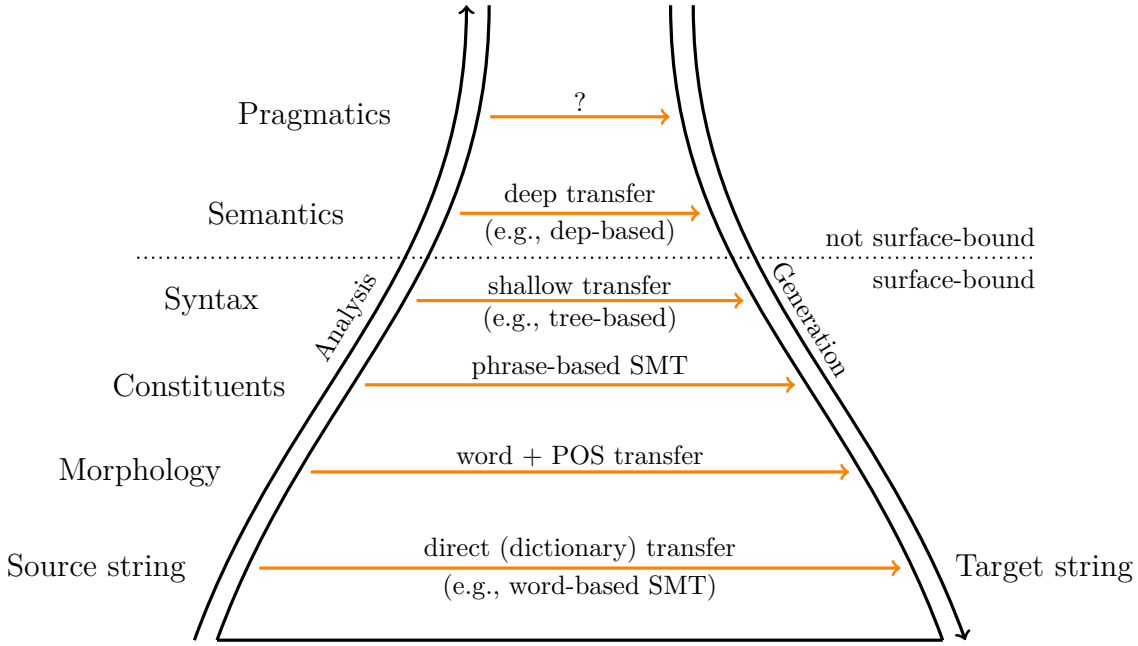


Figure 1.2: Vauquois inverted funnel with a very long spout

Using a higher-level representation, such as semantics, for transfer is theoretically appealing but there are practical challenges. One challenge is coverage. The methodology of semantic transfer that I use employs three grammars in the translation pipeline—a source grammar for analysis, a transfer grammar, and a target grammar for generation—and each

[3]See `http://courses.washington.edu/ling567/2017/0516.pdf`, retrieved 18 March 2018.

[4]See `https://www.cl.cam.ac.uk/teaching/0809/NLP/lectures.pdf`, retrieved 18 March 2018.

[5]See `http://compling.hss.ntu.edu.sg/courses/hg8003.2014/pdf/wk-08.pdf`, retrieved 18 March 2018.

grammar can fail to process some of its inputs, thus reducing the overall coverage at each step. Automatically extracting rather than hand-building transfer rules helps improve transfer coverage, but improving the source or target grammar coverage is not a goal of this dissertation. My evaluation compares the quality of items that were fully translated by computing automatic evaluation metrics over the intersection of translated items. A second challenge is disambiguation. As a sentence, the meaning of an input is implicit; that is, the sentence could correspond to many different interpretations. The analysis of a sentence to a syntactic or semantic representation must make decisions about particular interpretations, and thus the meaning of the input can become explicit but alternative meanings (perhaps even the intended one) can be lost in the process. My methodology addresses this challenge in two ways: (1) it uses a semantic representation that defers resolutions about ambiguity (e.g., lexical or quantificational) as much as possible, and (2) it enumerates the top N results (i.e., decisions) in case the first one is not the best. While a fully disambiguated analysis is the most informative for translation, for a data-driven model it increases data sparsity and increases the chances of choosing an incorrect disambiguation, so my methodology strikes a balance between expressiveness and learnability.

The tools and resources used by both low-level (e.g., phrase-based) and high-level (e.g., semantics-based) methods are possible because of language-specific information built into them. Machine translation systems, and NLP systems in general, often claim to be language-independent when they do not contain hard-coded decisions for a particular language, but language independence is better thought of as a scale than as a binary attribute, because there are many sources of linguistic information that are not coded decisions. A string-based method may rely on tokenizers, stemmers, lemmatizers, and text-normalizers that are developed or optimized for specific languages. The selection of a reordering model, and even the use of n-grams, embed linguistic biases into the system (Bender, 2011). Abstraction-based methods may also use the tools and methods above, but they also use representations that come from linguistic analysis of the languages and may take the form of annotated corpora, hand-built grammars, etc. My systems avail themselves of many sources of linguistic

information.

1.3 Main Contributions

This dissertation describes two new methods for extracting semantic transfer rules for rule-based machine translation, corresponding to two methods for bilingually aligning semantic subgraphs. The first method extends previous work by Haugereid and Bond (2011, 2012) on finding **predicate phrase pairs** by linearizing the source and target semantic predicates as if they were words (i.e., without any of the structure) then using an n-gram word aligner to find corresponding source and target n-grams. Haugereid and Bond (2011, 2012) matched these n-grams to templates and output stored semantic representations, but I avoid the use of hand-crafted templates by projecting the predicates onto the original semantic representations that were used for linearization. In this way, I recover the original, observed structure and avoid a potential deficiency where the structure stored in a template differs from the observed structure. The use of predicate n-grams, whether with templates or without, is an indirect way of finding aligned subgraphs, so in the second method I align subgraphs directly via graph traversals, inspired by previous work on both MRS (Jellinghaus, 2007) and LFG f-structures (Hearne and Way, 2003; Graham et al., 2009). I do this by enumerating the subgraphs for the source and target representations, pairing them via the Cartesian product, then applying several levels of graph filters and statistical filters to remove bad pairings. Both methods yield aligned semantic subgraphs, but do not offer a more granular bilingual mapping of individual nodes, which is important for producing high-quality transfer rules. Therefore, they both rely on a heuristic strategy for finding node alignments—a task I call **bilingual variable binding** since the nodes are mapped via the coindexation, or binding, of variable identifiers. This strategy finds an approximate, though possibly incomplete, mapping of node identifiers by binding those with the same node type (e.g., referential index or eventuality) and located in a similar position relative to the top node in the subgraph structure.

In order to define the methods of subgraph alignment and bilingual variable binding, I implemented several semantic operations for working with the MRS representations that

are produced and processed by the HPSG grammars. Some of these implementations are my own design, while others are adaptations of algorithms described elsewhere. The adaptations include MRS to DMRS (Copestake, 2009) conversion and deconversion and DMRS simplification. Novel implementations include a method of **structural isomorphism comparison**, implemented as a string comparision of a deterministic graph serialization; a test for **link orientation** based on link properties that is separate from the functor-argument direction of links; a **rooted traversal** that uses link orientation to convert a multiply-rooted DMRS graph to a singly-rooted graph; and two methods of DMRS subgraph extraction, one by matching n-grams of predicates and another via a depth-limited rooted traversal.

I use the above methods and operations to build two systems based on automatically extracted transfer rules. The first system extracts 359,407 aligned semantic subgraphs, leading to end-to-end translation coverage of 18.4%, a 5.8% increase over the previous state of the art for MRS transfer (Haugereid and Bond, 2011, 2012), which I use as a baseline, although at a slight drop in BLEU score (Papineni et al., 2001): 24.86 versus 30.40. The second system extracts 977,532 aligned semantic subgraphs, which is much more than my first system, but the end-to-end coverage is in fact lower: 10.6%, which is a 2% drop from the baseline. The BLEU score, however, is more competitive: 29.74 versus 30.40. My systems do not beat a Moses (Koehn et al., 2007) phrase-based baseline on BLEU (35.05), but they are competitive on the METEOR metric (Banerjee and Lavie, 2005; Lavie and Agarwal, 2007), where my second system gets 34.83 and Moses gets 35.80. In-depth analysis of my results shows that my extraction methodology is finding good aligned subgraph pairs but is (1) having difficulty fully utilizing the pairs in the transfer grammar due to an incomplete bilingual variable binding, and (2) possibly losing good analyses and transfers due to early filtering of translation hypotheses. These results show that my systems are of similar quality to MRS transfer baseline but, unlike the baseline, mine are produced without hand-built and hand-tuned templates, thus improving language independence and maintainability. The analysis further shows that there are several promising research directions that can improve the performance of the aligned subgraph pairs I extracted.

1.4 Document Outline

The rest of this dissertation is organized as follows. In Chapter 2 I cover DELPH-IN Semantics, the semantic framework that defines the representations I use as the medium of transfer in my machine translation experiments. Of particular relevance are Sections 2.6 and 2.10, which describe the MRS and DMRS representations, both of which are heavily utilized in my research. I situate my translation methodology in the broader literature in Chapter 3 where I describe the relevant paradigms and methods in prior machine translation research. The process of building a transfer model and using it to translate source sentences into target sentences is complex and involves many steps, so in Chapter 4 I provide a high-level overview of the transfer-grammar building and translation pipelines.

Chapter 5 is the first chapter describing my own methods and contributions, and covers the semantic operations I perform for transfer rule extraction, transfer grammar building, and translation. Section 5.1 defines the conversion of MRS into DMRS (Copestake, 2009), required for my methods of transfer rule extraction, and Section 5.2 defines the reverse procedure, converting DMRS into MRS, which is required for augmenting the transfer grammars. Section 5.6 explains my methods for extracting subgraphs from DMRS instances. Section 5.7 lists some graph transformations I apply to simplify rule extraction.

In Chapter 6 I explain my methods for bilingual semantic subgraph alignment—the first major step in extracting transfer rules from a bilingual corpus of semantic representations. I explore two different methods of alignment, each of which produce a **transfer pair store**. The first, described in Section 6.1, is the alignment of linearized semantic predicates, inspired by previous work by Haugereid and Bond (2011, 2012). The second, described in Section 6.2, is the direct alignment of semantic subgraphs enumerated via graph traversal, inspired partly by previous work within DELPH-IN (Jellinghaus, 2007) and for LFG representations (Hearne and Way, 2003; Graham et al., 2009). The methods require different forms of filtering to help ensure the resulting transfer pair stores only contain translationally equivalent mappings, but there are also filters common to both methods, described in Section 6.3.

Chapter 7 defines my methods for augmenting a core (i.e., minimal) transfer grammar with rules created from the extracted transfer subgraph pairs. I give an overview of the LOGON transfer machinery in Section 7.1. Section 7.2 describes how I select pairs from the transfer pair stores for a translation task, and how I order the pairs to improve the performance of the resulting transfer grammar. The subgraph pairs are converted to MRS transfer rules as described in Section 7.3, Each subgraph in a subgraph pair describes the relationships between its own nodes via edges and these relationships are encoded as coindexed variables in MRS. The process of binding variables (i.e., establishing coindexation) from node relationships is described in Section 7.3.1. The binding of variables bilingually, that is, encoding the relationships between source and target nodes, is more difficult than monolingual binding, but equally crucial for producing high-quality transfer rules. The subgraph pairs I extract do not contain any information about bilingual node relationships, so I use heuristics to approximate a useful bilingual binding, as described in Section 7.3.2.

Chapter 8 explores the data sources I use for building transfer grammars and for translation. The performance of data-driven methods, such as my transfer rule extraction, is heavily affected by the quality and quantity of the data given, so this exploration is useful for understanding what kinds of information my method can utilize and how it compares to that used by past research. I give qualitative descriptions of the corpora in Section 8.1 along with some examples from each. In Section 8.2 I give a quantitative analysis of the data, including the distribution of sentences by length (i.e., word count) and the number of duplicate entries. My translation pipeline works with the semantic representations produced by precision grammars, meaning it can only use data that is modeled by the grammars. In Section 8.3 I analyze the parsing performance of the Jacy (Siegel et al., 2016) and ERG (Flickinger, 2000) grammars, as I require the outputs of both for extracting transfer rules. In Section 8.4 I provide a similar analysis of generation performance, but only for the ERG as I only translate from Japanese to English, and thus I depend on the ERG's generation coverage. The intersection of semantic representations output by parsing both sides of a bitext results in a bilingual semantic corpus, or **bisem**, and I describe the bisem created

from my data sources in Section 8.5.

Chapter 9 describes my experimental design for evaluating the translation performance of transfer grammars produced via my rule extraction methods. I start by describing the evaluation metrics in Section 9.1, pipeline parameters in Section 9.2, and other data prepration steps in Section 9.3. I compare my systems to two baselines: a Moses (Koehn et al., 2007) system trained on the full bitexts, described in Section 9.4; and a transfer baseline based on the prior state of the art for rule extraction (Haugereid and Bond, 2011, 2012), described in Section 9.5. I have two experimental systems and for each I define a number of configurations for exploring the parameter space. The LPA system, described in Section 9.6, uses the subgraph pairs found via the linearized predicate alignment method. The SGA system, described in Section 9.7, uses the pairs found via enumerated subgraph alignment.

Chapter 10 reports the results of my experiments and in Chapter 11 I analyze those results. The results of LPA on the development data are given in Section 10.2 and those of SGA on the development data are given in Section 10.3. Section 10.4 reports the results over the test data, using the top-performing configurations from LPA and SGA. My experimental systems are competitive, but do not clearly improve on the baselines in the automatic quality estimations, so in my analysis I inspect numerous aspects of the systems to see where they do well and where they do not. I look at overlap (where a system produces the same translation as the reference sentence or the output of another system) in Section 11.1 and compare the systems' relative performance on automatic quality estimations in Section 11.2. In Sections 11.3 and 11.4 I look at graph properties such as the distribution of subgraph topologies and several well-formedness metrics for MRS. I also look at computational performance measures, such as the number of times a system hits a timeout or memory limit, in Section 11.5. In Section 11.6 I go through a number of examples and find linguistic constructions that are easy or difficult for my system to translate.

Finally I conclude in Chapter 12 with an overview of the contributions of this dissertation. I also list a number of directions for future research, including practical next-steps and more ambitious proposals.

Chapter 2

DELPH-IN SEMANTICS

A primary product of the DEep Linguistic Processing with HPSG INitiative (DELPH-IN) consortium[1] is the implemented HPSG grammars available for a variety of languages, including English (ERG; Flickinger, 2000), Japanese (Jacy; Siegel et al., 2016), Mandarin Chinese (Zhong; Fan et al., 2015), Indonesian (INDRA; Moeljadi et al., 2015), German (GG; Müller and Kasper, 2000; Crysmann, 2005), and more,[2] and one useful output of language analysis using these grammars is the semantic representation called **Minimal Recursion Semantics** (MRS; Copestake et al., 2005). In the semantic transfer methodology described in this dissertation, MRS is the object of transfer. I will explain how to inspect and modify MRS representations in Chapter 5 and how to operationalize MRS for transfer rule extraction in Chapters 6 and 7. Therefore in this chapter I give an overview of the representation, including its components and variant representations, so as to provide sufficient background for later discussions. An example will help to make the following discussion more concrete, so Fig. 2.1 shows a graphical visualization of an MRS for the sentence *The dog sleeps*. The components and interpretation of this visualization will be explained later in this chapter.

MRS is not the only semantic representation used within DELPH-IN, as a variety of derivative representations have been developed over the years. These derivatives include Robust Minimal Recursion Semantics (RMRS; Copestake, 2004), Elementary Dependency Structures (EDS; Oepen et al., 2004; Oepen and Lønning, 2006), DELPH-IN MRS Bilexical Dependencies (DM; Ivanova et al., 2012), and Dependency Minimal Recursion Semantics (DMRS; Copestake, 2009). MRS, however, continues to be the primary representation used

[1] `http://www.delph-in.net/`

[2] See `http://moin.delph-in.net/GrammarCatalogue` for a longer, but still incomplete list.

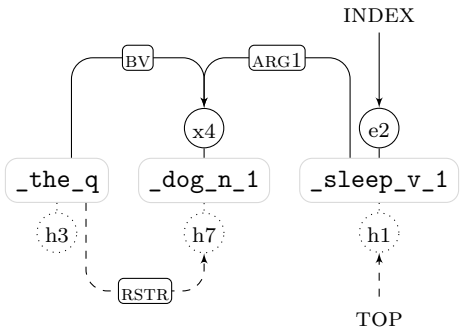


Figure 2.1: MRS graph view for *The dog sleeps.*

by DELPH-IN software and resources. The derivatives are often grouped under the heading of *MRS*, but I think this terminology confuses the more general semantic framework from the original MRS representation. Therefore, I will use MRS to refer to the representation as defined by Copestake et al. (2005), ***MRS** to refer to the family of MRS-based representations, and **DELPH-IN Semantics** to refer to the general semantic framework. Many of the principles of DELPH-IN Semantics were developed for MRS and do not necessarily persist in other *MRS representations. Nevertheless, those that I consider to be core parts of the framework are described below, while those specific to MRS are described in Section 2.6.

The remainder of the chapter is structured as follows. First I discuss the core philosophy and development desiderata of DELPH-IN Semantics in Section 2.1, and cover its basic structures and concepts in Section 2.2. I then explain the ways in which DELPH-IN Semantics is an underspecified representation in Section 2.3. Section 2.4 explains the visual representations of *MRS structures that I use in this dissertation. DELPH-IN Semantics are most commonly produced from HPSG grammars, so Section 2.5 explains the role of the SEM-I for interfacing between semantic representations and grammars. Sections 2.6 to 2.10 describe the various *MRS representations and I compare them to each other in Section 2.11. Finally, I compare DELPH-IN Semantics to some other semantic frameworks in Section 2.12.

2.1 Philosophical Basics

The term *semantics* has a broad range of interpretations, even when restricted to computational semantics, so here I will clarify what DELPH-IN Semantics is and what it is not. In a nutshell, DELPH-IN Semantics is a symbolic, structural, compositional, underspecified, and abstract representation of standing meaning. By **symbolic**, I mean that DELPH-IN Semantics is represented by discrete atomic symbols, such as predicates and role names, that stand in for some unknown real values, rather than continuous **numeric** values where meaning is, e.g., a point in some n-dimensional space. DELPH-IN Semantics is **structural** because it encodes the entities involved in an utterance and their relationship to one another; a representation that is not structural may, for instance, encode only lexical semantic information such as word-sense distinctions. Bender et al. (2015) explain how DELPH-IN Semantics is **compositional**, where the meaning representation for a sentence is monotonically built up from smaller meaning fragments that are each licensed by words in the sentence, and they compare this to AMR (Banarescu et al., 2013), a representation that is similarly symbolic, structural, and abstract, but is **non-compositional**. Bender et al. (2015) also discuss how compositionality relates to **standing meaning** (also called **sentence meaning**), where the semantic representation encodes only the information that can be obtained by the content of the linguistic signal; i.e., DELPH-IN Semantics is intentionally non-committal toward information that is not directly evidenced in the input. In contrast to standing meaning is **occasion meaning** (also called **speaker meaning**) which is the intended interpretation of the utterance and its intended effect on the addressee. DELPH-IN Semantics is **underspecified** because a single representation instance subsumes one or more logical analyses. See Section 2.3 for more discussion on underspecification. Finally, DELPH-IN Semantics is **abstract** in that it, like syntax, describes an utterance with relatively coarse-grained and generalized labels. In contrast, **concrete** semantic representations are those that are fully specialized for their domain-specific usage; e.g., SQL queries over a database.

DELPH-IN Semantics is not a semantic theory on its own, but a "meta-level language

for describing semantic structures in some underlying object language" (Copestake et al., 2005, pp. 282–283). While it, as a meta-level language, has high-level constraints on well-formedness, it is not defined with constraints on semantic validity tied to analyses for a particular language.[3] MRS was engineered to be composed in tandem with syntactic structures in frameworks like HPSG (Pollard and Sag, 1994) or LFG (Dalrymple, 2001),[4] leading to semantic structures that are close to the syntactic form, but which abstract away the semantically-empty manifestations of syntax (e.g., infinitival *to*, obligatory prepositions on arguments of certain verbal frames), and which are detached from the linear linguistic signal.

Early work on DELPH-IN Semantics was motivated by machine translation (Copestake et al., 1995) as part of the Verb*mobil* project (Wahlster, 1993; Bos et al., 1996), so its basic features were designed to accommodate translation via semantic transfer. Verb*mobil* was concerned with speech translation, and thus automatic speech recognition and discourse representations, but the development of DELPH-IN Semantics has mainly focused on complete textual sentences. Information from longer discourses or present in the audio signal could be useful for disambiguation or to create a richer representation, but these kinds of information are generally considered outside the purview of DELPH-IN Semantics. There are notable exceptions, such as work on resolving fragments by looking at structures in discourse (Schlangen and Lascarides, 2002), and in parsing fragments in dictionary definitions (Fujita et al., 2006). The compositional representation of standing meaning in, e.g., MRS, can be enriched and specified by processes external to composition when such information becomes known.

The main desiderata for the development of MRS, from Copestake et al. 2005, pp. 281–282, are:

Expressive Adequacy The framework must allow linguistic meanings to be

[3]By way of analogy, the XML data format has well-formedness criteria—e.g., regarding opening and closing tags, or character ranges—that are applicable to all XML documents, but the validity of a well-formed document is defined by a schema for a particular application.

[4]MRS has been shown to work with some non-lexicalist frameworks, like CFGs, as well (Copestake, 2007).

expressed correctly.

Grammatical Compatibility Semantic representations must be linked cleanly to other kinds of grammatical information (most notably syntax).

Computational Tractability It must be possible to process meanings and to check semantic equivalence efficiently and to express relationships between semantic representations straightforwardly.

Underspecifiability Semantic representations should allow underspecification (leaving semantic distinctions unresolved), in such a way as to allow flexible, monotonic resolution of such partial semantic representations.

In order to be expressively adequate, a grammar must be able to license structures that describe all possible interpretations of a sentence. The full enumeration of a sentence's possible scoped forms, however, could yield **trillions** of readings (Koller and Thater, 2010)—a computationally intractable number. Therefore MRS makes use of a mechanism, discussed in Section 2.3, for underspecifying quantifier scope while simultaneously, and necessarily, encoding predicates in a flat (i.e., **minimally-recursive**) list instead of a nested structure. By underspecifying quantifier scope, rather than being merely agnostic about it, MRS retains expressive adequacy while accommodating the criterion of computational tractability.

MRS, being a symbolic, structural, compositional, underspecified, and abstract representation of standing meaning, is found in a very specific corner in the space of possible *semantic* representations, but its position is one of general applicability and extensibility. It is these qualities that make it a useful representation for transfer-based machine translation.

2.2 Structural Semantic Preliminaries and Terminology

Here I briefly discuss the basic elements that are employed in DELPH-IN Semantic representations.

2.2.1 Predicate Symbols

Perhaps the most contentful elements in DELPH-IN Semantics are the predicates, as they encode the lexical material. A semantic representation containing only predicates would convey a lot more meaning than one containing only the argument structure, quantification scope, or other layers. In fact, this was the premise of a machine translation project called *lemmatic machine translation* (Soderland et al., 2009), which aimed to translate sentences between language pairs that have little or no parallel sentences, but do have a large bilingual dictionary. The results had high translation adequacy, but low fluency, meaning the important points were translated but evaluators had a hard time reconstructing how the points fit together (Everitt et al., 2010).

Semantic predicates in DELPH-IN Semantics can be cast into two broad categories: **surface predicates** include those corresponding directly to a surface token (e.g., `_dog_n_1` for the word *dog* or *dogs*, or `_bark_v_1` for the word *bark*, *barks*, *barked*), and **abstract predicates** include those that do not correspond directly to a surface token (e.g., `compound` for connecting the elements of a noun compound, `dofw` for abstracting over days of the week, etc.). Generally, the surface predicates are defined in the grammar's lexicon and are inserted into the semantic representation during parsing when a lexical entry is selected for an input token. Abstract predicates are typically inserted by grammar rules used in a parse, although they can also appear in the lexicon, as with `much-many_a`, which is used for the lexical entries for *much*, *many*, *lotta*, etc.

DELPH-IN Semantics takes a highly underspecified stance on lexical semantics, delegating the task of sense disambiguation largely to downstream processors. To adapt an old semantics joke, the meaning of *life* is `_life_n_1`.[5] More specifically, lexical senses that are not associated with differences in morphosyntax are conventionally not given distinct predi-

[5]This joke originates as the answer to a question in a class on Montague grammar. The class was co-taught by Barbara Partee and Terry Parsons at the University of Massachusetts in 1967 and they had offered to answer any question on the last day of class. Gregory Carlson, then a student, had asked, "what is the meaning of life?", to which Barbara Partee walked up to the chalk board and wrote `^life'`. A fuller account of this episode is given in the foreword of Gregory Carlson's dissertation (Carlson, 1977).

cate symbols in DELPH-IN grammars, as it would lead to otherwise uninformative ambiguity in analysis and thus run afoul of the goal of computational tractability, if not also the goal of underspecifiability (Copestake et al., 2005).

2.2.2 Predications and the Semantic Graph

An instance of a predicate and its outgoing arguments is called a **predication**. The resulting structure from the composition of a sentence's predications is the **semantic graph**, and is useful for encoding the difference between, for instance, *Dogs chase cats* and *Cats chase dogs*. Semantic graphs should be fully connected; when they are not, it is an indicator that something went wrong in parsing the sentence, such as a bug in a grammar rule. Arguments, or edges, have labels called **roles**, which are discussed in Section 2.3 with regards to underspecification.

2.2.3 Surface Alignments

Surface alignments are the correspondences between semantic material, like predicates, and positions in the linear linguistic signal (i.e., the surface form), like word or character indices. Other terms for surface alignments are **characterization** and **lnk values**, and often they are referred to by their feature names CFROM and CTO. While surface alignments aren't true semantic information, their utility justifies their (optional) inclusion in *MRS representations. For instance, they are used in the semantic similarity function Elementary Dependency Matching (EDM; Dridan and Oepen, 2011) to compare two semantic representations for the same input sentence. In this dissertation I use them for node iteration (see Section 5.4.1). I do not include surface alignments in transfer rules, and semantic subgraphs that differ only by their surface alignments are considered equivalent by my algorithms.

2.2.4 Morphosemantic Properties

Semantic predicates only encode lemmatic lexical content and certain kinds of sense distinctions, as described above, and not any morphological variations (and associated semantic contributions) present in the surface signal. A semantic representation would not encode the morphophonological processes or their effects on the surface form, but **morphosemantic properties** are useful pieces of information that encode, e.g., verb tense or nominal number. In MRS these are called **variable properties** because they are encoded on variables tied to predications, whereas in DMRS, a variable-free representation, they are encoded on the graph nodes. Morphosemantic properties are important for translation, as they are the only difference between sentences like *I called you* and *She had been calling them*, although in my dissertation I don't give them any special treatment.

2.3 Underspecified Representation

DELPH-IN Semantics is part of a family of **underspecified representations**, where a single representation instance subsumes one or more logical analyses, a family that includes Underspecified Discourse Representation Structures (Reyle, 1993) and Hole Semantics (Bos, 1996). DELPH-IN Semantics can be underspecified in a variety of ways. One is model-internal ambiguation, where an instance is underspecified over one or more more fully specified instances that can be represented by the semantic model of a grammar. This type of underspecification includes the use of predicates or property values that are non-terminal nodes in a predicate- or value-hierarchy, as well as the use of underspecified quantifier scope via handle constraints (discussed below). Another is model-external agnosticism, where the grammar engineer either decides to exclude some kinds of information from the semantic model, or is unaware of the availability of the information, i.e., the model is de facto underspecified because it has not been designed with the ability to make any such specification. This second type includes the use of bleached semantic role labels (e.g., ARG1 and ARG2 instead of AGENT and PATIENT), and course-grained lexical sense distinctions (also discussed

below).

One important form of underspecification in DELPH-IN Semantics is the encoding of quantifier scope ambiguity via **handle constraints**. Handle constraints describe the relationships between **handles**, which are labels identifying scopal positions. There are three relationships for handle constraints, **qeq**, **lheq**, and **outscopes**, but as qeq is the only one actively used in implemented grammars, I will not discuss the other two.[6] Qeq, or *equality-modulo-quantifiers*, also written $=_q$, allows for a partial ordering of quantifiers and other scopal predications in the scope tree. That is, if A is qeq B, then A must be above B in the scope tree, but it needn't be directly above (in which case one says A *immediately outscopes* B), and other quantifiers or scopal things could intervene between A and B. See Copestake et al. 2005 for a full account of how qeqs work, but for my research it is sufficient to say that qeqs play an important role in tractably encoding quantifier scope information in MRS representations, and that this information is necessary to make use of the standard machinery for surface realization from MRS forms (Carroll et al., 1999; Carroll and Oepen, 2005).

Examples (8a) and (8b) show the two scoped forms for the sentence *Every dog chases some cat.* (8a) is where every dog chases a cat but it may be a different cat for each dog, and (8b) is where there is some single cat that all dogs chase.

(8) a. $\mathsf{every}(x, \mathsf{dog}(x), \mathsf{some}(y, \mathsf{cat}(y), \mathsf{chase}(x, y)))$
b. $\mathsf{some}(y, \mathsf{cat}(y), \mathsf{every}(x, \mathsf{dog}(x), \mathsf{chase}(x, y)))$

MRS combines these two formulae into a single representation, as shown in (9), where it specifies that *dog* is in the restriction of *every* and *cat* is in the restriction of *some*, but does not say which quantifier has higher scope. Each predication gets a label, and the second bag in the structure encodes the relationships between handles.

(9) $\langle\, h_1, \{\, h_4{:}\mathsf{every}(x, h_7, h_5), h_8{:}\mathsf{dog}(x), h_{10}{:}\mathsf{some}(y, h_{12}, h_{11}), h_{13}{:}\mathsf{cat}(y), h_2{:}\mathsf{chase}(x, y) \,\}, \{\, h_1 =_q h_2, h_7 =_q h_8, h_{12} =_q h_{13} \,\} \,\rangle$

[6] I will, however, mention that using lheq is equivalent to simply coindexing the handles, a tactic that **is** employed by implemented grammars.

2.4 Visual Presentation of DELPH-IN Semantics

There are several semantic representations within DELPH-IN Semantics described in this chapter, and there are even more ways of encoding or visually presenting these representations. In (9) I used the inline form of the *indexed* view[7] for MRS, but I simplified and reordered the predicate symbols, simplified the variable names, and removed role labels, morphosemantic properties, and surface alignments so that it would be easier to compare to the logical formulae in (8a) and (8b). The full indexed version (9) is given in (10), which contains all the information encoded in the MRS.[8] When illustrating semantic phenomena, e.g., in academic papers, the morphosemantic properties and surface alignments are often omitted, and I will continue the practice in this dissertation.

$$
\begin{array}{ll}
(10) & \Bigg\langle\ h_1, e_3, \\
& \left\{ \begin{array}{l}
h_4{:}\_\mathsf{every\_q}\langle 0{:}5\rangle\big(\ \mathsf{ARG0}\ x_6\{\mathsf{PERS}\ 3, \mathsf{NUM}\ sg, \mathsf{IND}\ +\},\ \mathsf{RSTR}\ h_7,\ \mathsf{BODY}\ h_5\big), \\
h_8{:}\_\mathsf{dog\_n\_1}\langle 6{:}9\rangle\big(\ \mathsf{ARG0}\ x_6\big), \\
h_2{:}\_\mathsf{chase\_v\_1}\langle 10{:}16\rangle\left(\begin{array}{l} \mathsf{ARG0}\ e_3\{\mathsf{SF}\ prop, \mathsf{TENSE}\ past, \mathsf{MOOD}\ indicative, \mathsf{PROG}\ -, \mathsf{PERF}\ -\}, \\ \mathsf{ARG1}\ x_6, \\ \mathsf{ARG2}\ x_9\{\mathsf{PERS}\ 3, \mathsf{NUM}\ sg, \mathsf{IND}\ +\} \end{array} \right), \\
h_{10}{:}\_\mathsf{some\_q\_indiv}\langle 17{:}21\rangle\big(\ \mathsf{ARG0}\ x_9,\ \mathsf{RSTR}\ h_{12},\ \mathsf{BODY}\ h_{11}\big), \\
h_{13}{:}\_\mathsf{cat\_n\_1}\langle 22{:}26\rangle\big(\ \mathsf{ARG0}\ x_9\big)
\end{array} \right\}, \\
& \{h_{12} =_q h_{13}, h_7 =_q h_8, h_1 =_q h_2\}
\end{array}
\Bigg\rangle
$$

The indexed view is a standard way of presenting MRS structures in papers, and while it has its proponents, I find the task of mentally assembling the semantic structure by locating coindexed variables too cumbersome. In this dissertation I use an alternative dependency-like graph view that allows one to more readily grasp the overall structure as well as see interesting substructures. The indexed MRS in (10) is presented graphically in Fig. 2.2. Note that, in addition to omitting the morphosemantic properties and surface alignments, I also drop unused arguments (e.g., the BODY role on quantifiers) and variables that are

[7]The documentation for the ERG's semantic analyses (`http://moin.delph-in.net/ErgSemantics`) call this the *Simple MRS* view, but I find this name misleading because there is already a data format for MRS with the same name, along with a similar presentation format. In addition, there is a data format called *Indexed MRS* that is similar to the presentation format I use in this dissertation.

[8]There are in fact a few other pieces of information that **can** be encoded in an MRS, such as the surface form of the sentence or of a predication, but these are not often given by grammar processors.

uninformative in a graphical view (e.g., the TOP, INDEX, and hole variables, as their presence and position are entirely predictable from the edges they are part of).

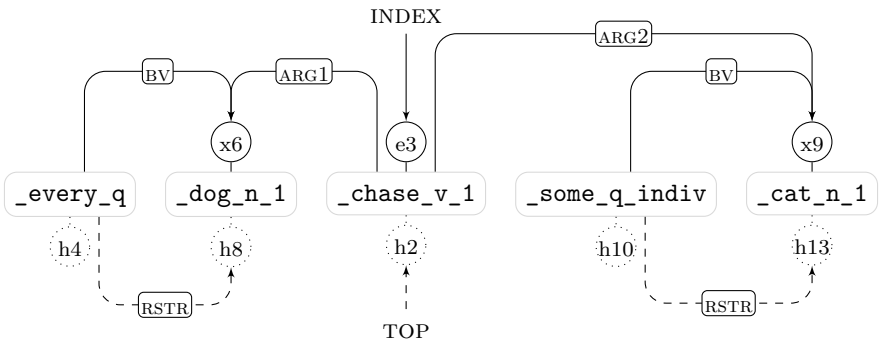


Figure 2.2: MRS graph view for *Every dog chased some cat.*

The components of the view require some description. Predicates are arranged on a horizontal axis in the order output by the grammar processor, which is approximately the surface order of the words that licensed the predicates. Labeled edges connecting predicates to variables are role arguments, and those above the predicates are non-scopal while those below are scopal. Non-scopal edges are solid lines (→) and point to the intrinsic variable of the target predicate, which is given as a variable in a circle ((e3)). Scopal arguments are dashed lines (- →) for qeqs and solid lines (→) when the argument selects a label directly (i.e., where the source immediately outscopes the target). Dotted edges (⋯) select one or more predicates that are in a scope conjunction, with the scope label given as a handle in a dotted circle ((h2)). Finally, the INDEX and TOP edges point to the top variable and top label, respectively.

2.5 SEM-I: The Semantic Interface

Before I describe the various representations of DELPH-IN Semantics below, I will explain another part of the semantic framework that is useful for consumers of the representations: the **semantic interface** (SEM-I; Flickinger et al., 2005b). Some semantic representations

link their predicate or concept inventory to an external lexical database, as AMR (Banarescu et al., 2013) does with OntoNotes (Pradhan et al., 2007) (see Section 2.12), but DELPH-IN Semantics are built on an inventory internal to the grammar used to produce the representations. A broad-coverage, deep grammar like the ERG (Flickinger, 2000) is a complex system of many parts (for details see Section 4.3) with a vast collection of lexical information. The semantic representations produced by a grammar are consistent with its model, with all lexical predicates being defined by in the lexicon or the type hierarchy. If a downstream application had to load the entire grammar in order to understand the semantic representations, it would be non-trivially burdened by the complexity and weight of the grammar. In order to alleviate this burden, the information relevant to the semantic representations is extracted from the grammars and put into a structure called a SEM-I.

A SEM-I contains descriptions of the elements in a semantic representation and their range of values. It describes the possible variable types and the hierarchical relationships between them; the common, if not ubiquitous, variable hierarchy used in DELPH-IN grammars is illustrated by Fig. 2.3. The `e` variable is for *eventualities*, e.g., events and states, as from verbs or adjectives. The `x` variable is for *referential indices*, such as from regular nouns, pronouns, nominalized verbs, etc. The `h` variable is for handles, i.e., for identifying scopal positions. Above these three basic variable types are their underspecified supertypes, which are useful in a variety of situations. For instance, the ERG only requires one predicate for two senses of *believe*, as in *Kim believes Sandy*, or *Kim believes that Sandy slept*, as the second argument is constrained to be of type `p`, i.e., the supertype of a referential index (like *Sandy*) or of a handle (like that which identifies the scope of *Sandy slept*).

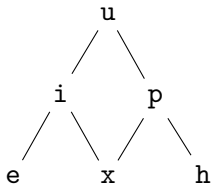


Figure 2.3: Variable hierarchy

In addition to variable types, a SEM-I also encodes what morphosemantic properties can be associated with variables of these types, and the range and hierarchical arrangement of these property values. The allowed semantic roles are given with their most general constraints on argument types. Finally, a SEM-I encodes the grammar-defined semantic predicates defined with their supertypes (if the predicate exists in a hierarchy), arity, more specific constraints on argument types, and constraints on morphosemantic properties.

As a SEM-I is an **interface** into the semantic framework defined by the grammar, meant for grammar-external consumers, there may be situations where a SEM-I can shield the consumer from irrelevant implementational details or provide abstractions that the grammar itself does not make. For example, in the 1214 version of the ERG, its SEM-I defines `existential_q` as a more user-friendly alias of `def_udef_some_a_q`. Beyond just user-friendliness, these aliases also help keep the interface more consistent, as `def_udef_some_a_q` is renamed to `def_udef_some_a_no_q` in a recent version of the ERG. This change would be problematic for applications that targeted the original name, but not for those that used `existential_q`. A second example is an abstraction I added to the ERG for Japanese-English translation with the JaEn grammar.[9] The `existential_q` quantifier abstraction is useful when translating from Japanese, which does not have article determiners like *a* or *the* in English, but it is too abstract and may be realized as a demostrative (e.g., *this* or *those*). Japanese does have overt demonstratives, so these translations would be inappropriate when no demonstrative existed in the Japanese sentence. Therefore I insert in the ERG's quantifier hierarchy a predicate `def_udef_a_q` that captures only *a*, *the*, and null quantifiers (allowed in English when the quantifiee is plural, as in *Dogs bark*).[10] The target grammar, upon encountering `def_udef_a_q` from, e.g., the JaEn transfer grammar, realizes a quantifier from the more restricted set, leading to more accurate translations.

Modifications to a SEM-I such as these can make life easier for downstream processors of DELPH-IN Semantics (i.e., consumers of grammar-defined semantic models, such as the

[9]Thanks to Stephan Oepen and Francis Bond for the solution.

[10]The details of the implementation are given in Appendix A.2.

above which makes a convenient sub-class of quantifiers available for JaEn to use), but they also carry risks. They often solve a specific problem, so there may be a limited range to their utility and applicability. As they change the semantic model for a specific version of the grammar, they have a limited life span. In my case, I altered the ERG's semantic model to address a specific deficiency for transferring Jacy semantic representations to representations for the 1214 version of the ERG, but these modifications are not necessarily useful for other tasks that use the ERG and the changes may become outdated with future versions of the grammar.

Above I have described a SEM-I as an interface that helps downstream consumers of semantic representations get at the semantic model defined by a grammar without having to load and interpret the entire grammar. One could imagine extending it to map the DELPH-IN Semantics model to external ontologies, as well. Roa et al. (2008), for instance, describe a statistical technique for mapping more fine-grained semantic role labels from PropBank (Palmer et al., 2005) or VerbNet (Kipper-Schuler, 2005), such as *Experiencer* or *Theme*, to the bleached roles used in MRS. I only use a SEM-I for modifying the predicate hierarchy, as described above, to assist transfer when the MRS semantic models for two grammars do not have translationally equivalent predicate hierarchies. Next I will discuss the structure of MRS instances.

2.6 MRS: Minimal Recursion Semantics

Minimal Recursion Semantics (Copestake et al., 2005) is the original semantic representation in DELPH-IN Semantics, and the progenitor of the succeeding representations. The following is a brief description of MRS, adapted to modern usage and assumptions about the representation. Each MRS instance is a tuple $\langle gt, i, R, HC, IC \rangle$. gt is the global top handle; that is, no other handle may scope over gt. i is the top index, i.e., the intrinsic variable of the main predication. i was not included in the original definition of an MRS structure, but its usage is near-ubiquitous among modern DELPH-IN grammars, and therefore is included in this definition. It often, but not always, points to the same EP as gt, and differs in cases

where the outermost scope is not represented by the top EP in terms of argument structure, as in *The dog probably barks*, where *probably* is the scopal top but *barks* is the index, or when the top scope captures more than one EP, as in *The dog barks loudly*, where both *barks* and *loudly* share the top scope, but again only *barks* is selected by the index. R is a bag of predications (**elementary predications**, or EPs). HC is a possibly empty bag of **handle constraints** (HCONS), which are used to describe the scope tree. IC is a possibly empty bag of **individual constraints** (ICONS), which are used to encode information structure, among other things (Song and Bender, 2011; Song, 2014). Like i, IC also was not included in the original definition, but unlike i it is relatively new and is thus employed by few grammars, with the English Resource Grammar (Flickinger, 2000) being a notable exception.[11]

Elementary predications have their own internal structure described by the tuple $\langle p, l, v, A\rangle$. p is a **semantic predicate** (also called a **relation**). l is a scopal handle (called its **label**). v is the **intrinsic variable** (at times called the **distinguished variable**, **characteristic variable**, or ARG0 after the role the variable is the argument of). The label and intrinsic variable at the EP level correspond to the TOP and INDEX at the MRS level. A is a possibly empty list of arguments, which may be non-scopal (the argument is a non-handle variable that is the intrinsic variable of some other EP, or an underspecified variable if it is a dropped argument), scopal (the argument is a handle which selects, directly or via qeq, one or more EPs sharing a scope), or constant arguments (proper names, numbers, etc.). In the original definition, the list of non-scopal arguments ("ordinary variable arguments" (Copestake et al., 2005)) was separate from the list of scopal arguments, constant arguments are not mentioned at all, and the intrinsic variable v had no special status. In the modern convention, however, it is assumed that every EP has an intrinsic variable and each intrinsic variable is unique to a single EP (excepting quantifiers, which do not have intrinsic variables). This convention is called the **intrinsic variable property**. Prior to this convention, it was standard for grammars to have linguistic analyses where some EPs have no intrinsic variable or where

[11]Currently the latest development version of the ERG employs ICONS, but they are not included in the `1214` version that I use for my experiments.

multiple non-quantifier EPs share an intrinsic variable. These deviations lead to undesirable interactions in processes that assume an MRS will have a particular shape, such as DMRS conversion, and so recent efforts have tried to harmonize grammars to adhere to the intrinsic variable property. While the ERG has been updated to the new convention, work is ongoing for other grammars, such as Jacy (Siegel et al., 2016).

Handle constraints and individual constraints are simple triples ⟨*lhs*, *rel*, *rhs*⟩. For HCONS, *lhs* (given by the feature HI, or HARG) is a handle-valued slot (also called a **hole**) selected by the scopal argument of some EP; *rel* is the relation of the constraint, which is one of `qeq`, `lheq`, or `outscopes` (see Section 2.3 for discussion on why I only consider `qeq` relations); and *rhs* (given by the feature LO or LARG) is the label (handle) of some other EP. I will not consider ICONS in this thesis, but they differ from HCONS in that *lhs* and *rhs* are instance variables (e.g., intrinsic variables, not handles), and *rel* can be any pertinent relation defined by a grammar.

A semantic algebra for MRS was created to ensure that semantic composition in a constraint-based grammar produces well-formed MRSs (Copestake et al., 2001). However, there is no systematic way to check a grammar for algebra compliance, so it's possible for the grammars to output ill-formed MRS outputs.

Figures 2.4a to 2.4e show the MRS structure of some fragments of varying complexity for the sentence *the large and gentle dog sleeps*, as parsed by the 1214 version of ERG. Fig. 2.4b shows the MRS structure for *The dog*, illustrating how the quantifier selects both the index (via the BV (bound variable) role) and the label (via a qeq from the RSTR role) of *dog*. Fig. 2.4d shows the MRS for *The large dog*, expanding on the previous MRS with non-scopal modification and illustrating how the RSTR role selects a label that is shared by more than one predicate, but the BV role still selects the single quantifiee. Fig. 2.4e expands on the example yet again, showing coordinated modifiers in the context of the full sentence. In this case, the non-scopal modifiers *large* and *gentle* select the modifiee, but they do not share a label. They are unable to share the label with the modifiee because the coordinator *and* shares its label with *dog* and immediately outscopes its coordinands, so it would be ill-formed

for the modifiers to share a scope with the thing that outscopes them. Also note that the `_and_c` has no direct edge with `_dog_n_1`.

2.7 RMRS: Robust Minimal Recursion Semantics

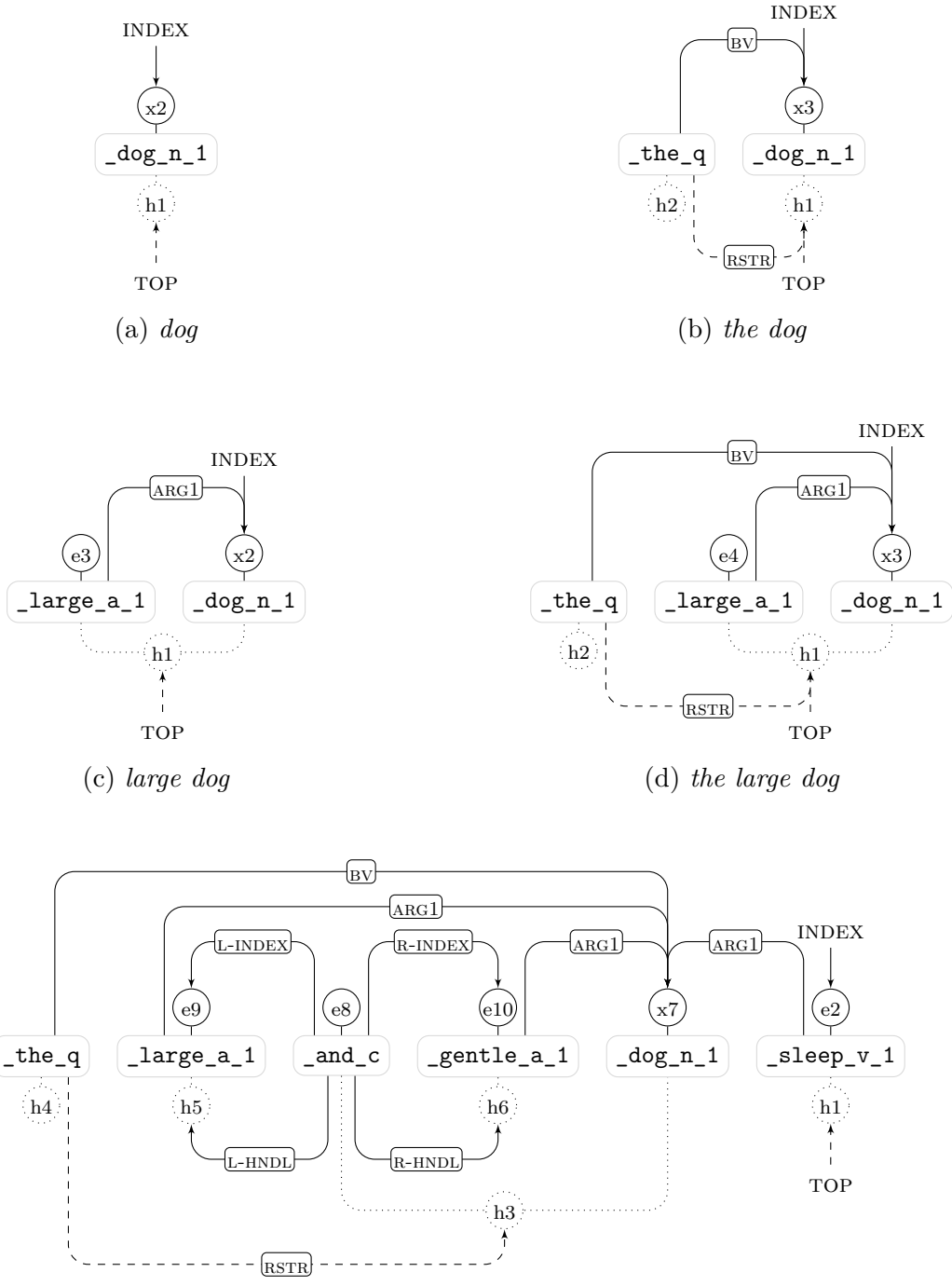
A common criticism of rule-based grammars like those developed within DELPH-IN is that they are brittle, failing to produce analyses for some inputs, and that they are slow to arrive at an analysis. An effort to combine the strengths of shallow processing techniques, like tagging, with deep grammars led to the development of Robust Minimal Recursion Semantics (RMRS; Copestake, 2004), which allows for underspecification of predicate symbols, argument roles, and morphosemantic properties. RMRS also represents EPs in a Parsons-style notation (Parsons, 1990) where each role argument is given first-class status, linked to their respective EPs via a unique **anchor** variable.[12] These anchor variables are an important advancement, as they give each EP, including quantifiers, a unique identifier, which is something that the intrinsic variables were unable to accomplish as they were not originally an obligatory part of the formalism.

Outside of a few applications that utilized its capability for integrating shallow processing techniques with deep grammars, such as DeepThought (Callmeier et al., 2004) and Heart of Gold (Schäfer, 2007), RMRS did not really take off as a semantic representation, and I will therefore not discuss it further in this dissertation.

2.8 EDS: Elementary Dependency Structures

The interpretation of MRS and RMRS representations can be computationally-intensive, enough to preclude their use for things like semantic search applications, and the complexity of the representations can make them cumbersome to work with for other tasks. Around the same time as the development of RMRS, Oepen et al. (2002; see also Oepen and Lønning 2006) developed the first variable-free dependency representation of MRS called as

[12]For Parsons (1990), the anchor variable was instead an *event variable*.



(a) *dog*

(b) *the dog*

(c) *large dog*

(d) *the large dog*

(e) *the large and gentle dog sleeps*

Figure 2.4: MRS fragments of *the large and gentle dog sleeps*

Elementary Dependency Structures (EDS). Conversion to EDS is lossy, as it removes scope information in favor of representational simplicity. EDS is intended not to be a fully expressive representation, but to facilitate downstream tasks like information retrieval (Kouylekov and Oepen, 2014) or internal tasks like parse disambiguation (Toutanova et al., 2005). Like RMRS, EDS makes use of unique identifiers for each EP, but unlike RMRS, these identifiers are simple strings and not variables. In order to encode the relationships from scopal arguments (which are no longer scopal in EDS), the concept of a **representative node** was created which uses a set of heuristics to select the most salient node from the set of nodes that formerly shared a scope. These heuristics make use of features based on the graph topology as well as some grammar-specific disambiguators. These grammar-specific features have been implemented for the ERG, but if other grammars want to fully utilize the capabilities of EDS, they will need to define interface functions to allow the conversion process to make use of information in the grammar.

2.9 DM: Bilexical Dependencies

Some users of DELPH-IN Semantics may desire the semantic structure to directly annotate the input string rather than working with a detached representation. To satisfy those users, DELPH-IN MRS Bilexical Dependencies (DM; Ivanova et al., 2012) further reduce EDS to project the dependency graph directly onto the surface tokens. Conversion to DM makes use of the features of EDS, such as the representative node selection, but also grants representations for tokens in the original sentence that are not included in other *MRS varieties, such as the infinitival *to* or obligatory prepositions, although these additional nodes are disconnected from the primary semantic graph. Since the source and target nodes of the dependencies in DM are always surface tokens, predicates (both surface and abstract) are not used. Abstract predicates in the source EDS that represent a binary relation, such as `compound`, may be encoded as edge labels in the DM. Other abstract predicates from the EDS may be dropped entirely, unless there is no other node label for the token they annotate.

Note that DM is surface-bound; i.e., it reattaches the semantic structure to the linear

linguistic signal. However, while the nodes are bound to the surface tokens, they are a dependency representation, and as such the edges are free to connect discontiguous (i.e., non-adjacent) semantic dependencies. This distinguishes them from other representations, like syntax trees.

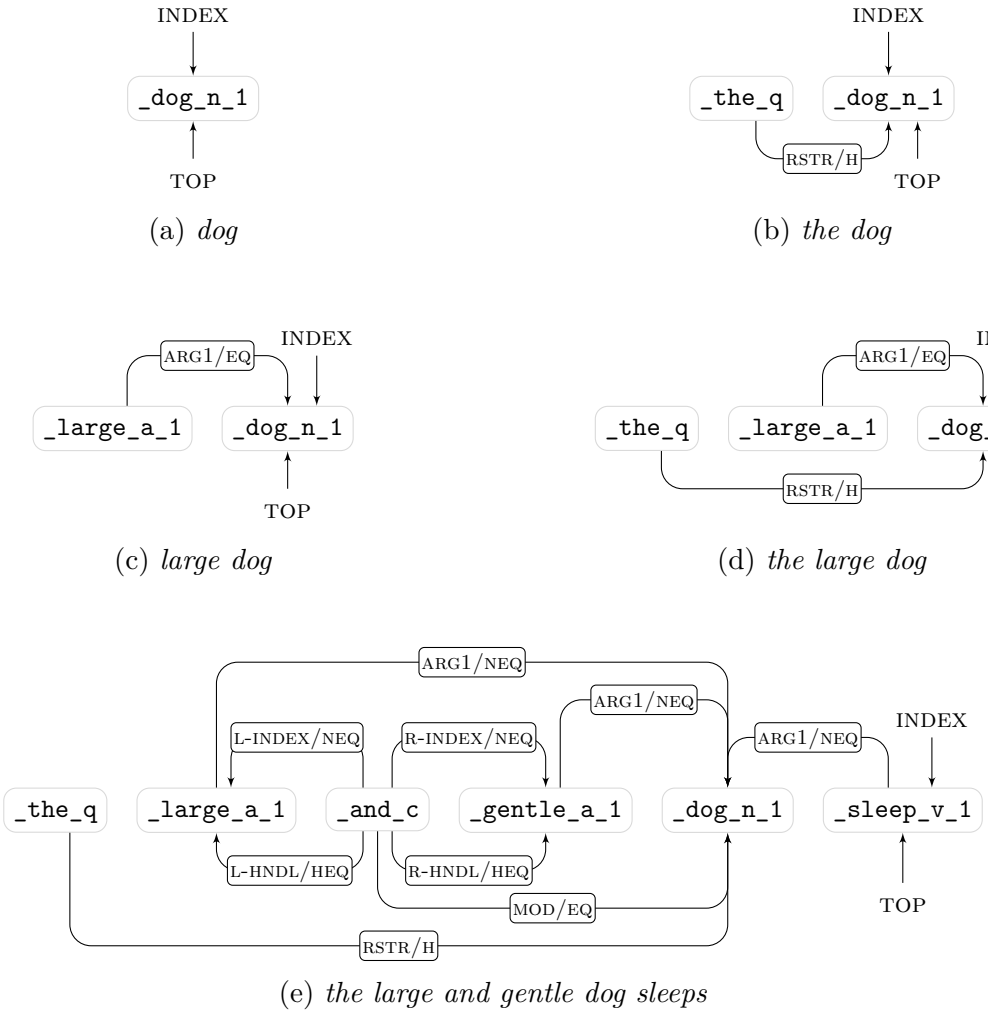
2.10 DMRS: Dependency Minimal Recursion Semantics

Dependency representations of MRS are easier to work with than MRS, but the lossy nature of EDS and DM makes it difficult to map back to the grammars in order to, for instance, realize sentences from the semantics. Copestake (2009), therefore, created Dependency Minimal Recursion Semantics (DMRS) as a dependency representation that maintains scope information as edge attributes. It uses features from EDS, such as the selection of representative nodes for EPs that share a label in the MRS representation. Chronologically it was developed after EDS, so one could say that DMRS is an extension of EDS, although they followed separate development tracks. In terms of the information encoded, however, it is a superset of EDS and a subset of MRS, so one could also say that EDS is a reduced form of DMRS. For a constrained class of MRSs, conversion to DMRS is lossless. DMRS does not represent the roles of dropped arguments, so for example the MRS representation for *Kim gives to charity* would have an underspecified ARG2 for the thing that is given (e.g., money, time), but in DMRS it is missing entirely. This is not considered a lossy conversion because the role's absence is interpreted as being the same as a having an underspecified argument, and the same sentences would be generated for both representations.[13] Furthermore, the

[13]One MRS that would have a lossy conversion to DMRS involves multiple coindexed dropped arguments, such as for the Japanese sentence in (ii). This sentence involves coordinated VPs with a shared, dropped subject. In the MRS, the missing ARG1 of both 進む *susumu* "advance" and 打つ *utsu* "hit" share the same variable even though no EP has that variable as its intrinsic variable. In DMRS, the ARG1 roles are not represented by any links, so the association is lost.

(ii) ボール を 進ん で 打つ
booru -wo susun -de utsu
ball -ACC advance -INF hit
"Step forward and hit the ball." [jpn]

missing arguments could be recovered by inspecting the predicate inventory of the SEM-I. More details on the MRS → DMRS conversion process are given in Section 5.1. Figures 2.5a to 2.5e show DMRS fragments for the same examples as Figures 2.4a to 2.4e do in Section 2.6, adapting the graph-view visualization for DMRS structures.



(a) *dog* (b) *the dog*

(c) *large dog* (d) *the large dog*

(e) *the large and gentle dog sleeps*

Figure 2.5: DMRS fragments of *the large and gentle dog sleeps*

Besides the encoding of scope information, DMRS contains nearly the same information as EDS. They are both variable-free representations, with morphosemantic properties encoded on nodes rather than variables. Unlike EDS, however, DMRS retains the sort (or type) of the

intrinsic variable along with the node attributes.[14] The retention of the scope information allows a DMRS graph to be considered connected for MRS instances where two EPs share only a label, a situation that is represented with a MOD/EQ edge (as in Fig. 2.5e).[15] Due to the greater recoverability of MRS structures, necessary for generation, from DMRS than from EDS, I chose to use DMRS as the representation used in the extraction of transfer rules.

2.11 Summary of DELPH-IN Representations

There are a variety of semantic representations in the DELPH-IN ecosystem, each conceived and designed for a different use case—MRS for transfer-based MT, RMRS for integrating shallow parsing techniques, EDS for facilitating downstream processing tasks like parse ranking or information extraction, DM for increasing the accessibility of DELPH-IN resources to the broader community, and DMRS for extending dependency representations with scope information—but they all ultimately derive from the original MRS representation, so they share many of the same qualities. Table 2.1 presents a feature matrix of the kinds of information that are present in the *MRS variants described in this section. The presence of the information is not always a simple yes or no answer, so where necessary I provide notes to explain the situations. The representations are arranged in the order of decreasing information density. Fig. 2.6 puts the representations (except RMRS) on an axis showing what major features are lost (or gained).

2.12 Comparison to Other Frameworks

There are many semantic representations beyond DELPH-IN Semantics and in this section I will briefly compare some of them to *MRS representations. Generally, the representations listed below influenced or inspired my own research.

---

[14]The intrinsic variable type is generally recoverable from the SEM-I, but some predicates can differ. For instance, color names in the ERG have `x` variables when they are nouns (*yellow is my favorite color*) and `e` variables when they are adjectives (*the yellow sun*).

[15]In Copestake 2009 these were called undirected /EQ edges. More recent versions of DMRS use the directed variant.

Table 2.1: Feature matrix of the prominent *MRS variants

	top	index	surface predicates	abstract predicates	EP/node identifiers	non-scopal arguments	scopal arguments	morphosemantic properties	constant values	quantifier scope	variables	hcons	icons
MRS	✓	✓	✓	✓	( )[a]	✓	✓	✓	✓	✓	✓	✓	✓
RMRS	✓	✓	✓	✓	var	✓	✓	✓	✓	✓	✓	✓	
DMRS	✓[b]	( )[c]	✓	✓	int	✓	✓[d,e]	✓[f]	✓[g]	✓		( )[d,e]	( )[h]
EDS	✓		✓	✓	str	✓	✓[d,i]	✓[f]	✓[g]			( )[d,i]	
DM	✓			( )[j]	int	✓	✓[d,i]		( )[k]			( )[d,i]	

[a] intrinsic variables do not always uniquely select nodes; see discussion in text
[b] implicitly encoded by linking special nodeid 0 to top
[c] an extension encodes INDEX as a graph attribute
[d] representative node of scopal set is chosen as sole target
[e] scopal relationship encoded on edge label
[f] properties exist on nodes rather than variables
[g] constant values exist as properties of nodes rather than pseudo-arguments
[h] support is planned but is not yet fully defined or implemented
[i] scopal relationship is dropped; arguments become non-scopal
[j] non-lexical abstract predicates (e.g., for compounding or subordination) are encoded as edge labels; others (e.g., implicit quantifiers) are dropped
[k] constants are irrelevant as the surface tokens are part of the node labels

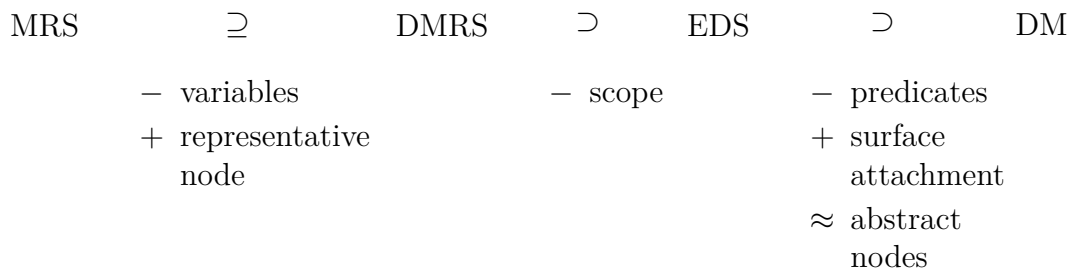


Figure 2.6: Ordering of information density among MRS, DMRS, EDS, and DM

Abstract Meaning Representation (AMR; Banarescu et al., 2013) is a representation that has grown in popularity over the past five years, although it was introduced much earlier (Langkilde and Knight, 1998), building off of earlier work in the PENMAN project (Penman, 1989; Kasper, 1989). While it, like DELPH-IN Semantics, encodes abstract structural semantics of sentences, its method of annotation is very different. The concept (i.e., predicate) inventory for AMR comes from OntoNotes (Pradhan et al., 2007) where possible. AMR corpora are manually annotated in the spirit of the syntactic treebanks of the 1990s, such as the Penn Treebank (Marcus et al., 1993). Annotators use their intuition to resolve coreference and ambiguity, resulting in a single representation for a sentence. A set of guidelines (AMR, 2017) instruct the annotators on best practice. For example, annotators are encouraged to decompose nouns into a verb and generic noun construction, where possible. *Teacher* may be annotated by a graph meaning *person who teaches*, but, in contrast, the decomposition of *professor* does not usually mean the same thing as the composed form, so it is left as is. But it is not always obvious when one should or should not decompose things. For instance, *spy* is not decomposed as *person who spies*, but *opinion* is decomposed to *thing that is opined*. Quantifiers are generally not represented, nor is any distinction of scopal or non-scopal argumentation. As such, negation is represented simply as a polarity change on a single node. The most recent release of the main AMR dataset contains representations for 39,260 English sentences (Knight et al., 2017). The PENMAN format used by AMR was the inspiration for my use of the same format for singly-rooted DMRS structures (see Section 5.5).

Discourse Representation Theory (DRT; Kamp et al., 2011) uses a compositional representation (called **Discourse Representation Structures**, or DRS) that is concerned with accurately expressing meaning in context, e.g., within a discourse. DRS is similar to MRS in that it uses variables for events.[16] The Groningen Meaning Bank (GMB; Bos et al.,

[16]Technically, MRS is a Davidsonian representation because semantic roles are associated with the event's predication, whereas DRS is neo-Davidsonian because roles are associated with the event. RMRS, however, is similar to DRS in this way.

2017) is the associated resource, containing representations for 10,000 open-domain texts.[17] Boxer (Bos, 2015) is a tool that produces DRS from CCG (Bos et al., 2004; Steedman and Baldridge, 2011) derivations. The compositional and event-based semantics may make DRS a promising target framework for future adaptations of my methodology.

Lexical Functional Grammar (LFG; Dalrymple, 2001), like HPSG, is a syntactic framework, and its **f-structure** is a deep representation of syntax that is used, for instance, as the medium of transfer in machine translation (Hearne and Way, 2003; Graham et al., 2009). F-structure encodes syntactic relationships but is abstracted (that is, *detached*) from the surface word order. This makes it very similar to semantic representations in some ways, but it, in contrast, models syntactic features such as subjects and objects instead of semantic roles (e.g., agents, patients, or themes; or the bleached roles found in MRS: ARG1, ARG2). LFG is also used as means of syntactic composition (as is usually done by grammars in the HPSG framework) of MRS semantics, as for NorGram (Dyvik et al., 1999; Rosén et al., 2005). The second of my methods for bilingually aligning semantic subgraphs (see Section 6.2) is partially inspired by machine translation methods developed within the LFG community.

2.13 Chapter Summary

In this chapter I have given an overview of DELPH-IN Semantics—the representation I operationalize for my research into the automatic extraction of machine translation transfer rules. I described DELPH-IN Semantics as a symbolic, structural, compositional, underspecified, and abstract representation of standing meaning (Sections 2.1 to 2.3), which are useful properties for general-purpose semantic modeling and bilingual transfer. In Section 2.4, I explained the visual representation of MRS that I use in this dissertation. The source of semantics I use for extracting transfer rules come from implemented grammars of Japanese and English, and in Section 2.5 I briefly explain how the SEM-I provides an interface into the grammars' respective semantic models. I describe the canonical MRS representation in Sec-

[17]Sentence counts are not available, but the 10,000 texts contain roughly 1.4 million tokens.

tion 2.6, followed by the derivative representations RMRS (Section 2.7), EDS (Section 2.8), DM (Section 2.9), and DMRS (Section 2.10). In Section 2.12, I briefly cover semantic representations outside of DELPH-IN Semantics. In this dissertation, I primarily work with the DMRS representation (e.g., in Chapter 5), but the grammars produce and consume MRS, e.g., for transfer (see Section 7.1) so I make use of both representations.

Chapter 3
MACHINE TRANSLATION

Machine translation (MT) is one of the oldest subfields of computational linguistics with a long and diverse literature but here I will provide a short overview of the aspects of MT relevant to my dissertation. My work exists within the rule-based machine translation (RBMT) paradigm but it also includes aspects of example-based machine translation (EBMT) and statistical machine translation (SMT).

3.1 Rule-based Machine Translation

Rule-based machine translation (RBMT) in the DELPH-IN[1] tradition begins with the Verb*mobil* project (Wahlster, 2000), which pursued a method of **semantic transfer** that rewrote source semantic inputs to target semantic outputs. The LOGON project (Lønning et al., 2004) continued and expanded this idea of semantic transfer with support for rule type hierarchies and a chart-based search for the transfer process. For these early systems, the full translation pipeline, from analysis through transfer to generation, was constructed with hand-built components. Statistical reranking models were applied to each pipeline component and as an end-to-end reranker (Oepen et al., 2007; Velldal, 2008), which helped bridge the divide between the rich, linguistically motivated representations of rule based systems and the data-driven selectional models of statistical systems. Fully hand-built transfer grammars are slow to develop and costly to maintain, so Jellinghaus (2007) devised a method of automatically acquiring MRS transfer rules using a top-down argument crawling method and Nichols et al. (2007) created transfer rules from bilingual dictionaries. Later work applied statistical word alignment tools to automatically extract transfer rules from parallel corpora (Haugereid and

[1]DEep Linguistic Processing with HPSG INitiative: `http://www.delph-in.net/`

Bond, 2011, 2012; Bond et al., 2011). One of my transfer pair extraction methods builds on Haugereid and Bond (2011, 2012) and is described in Section 6.1.

Well before DELPH-IN-style RBMT, example-based machine translation (EBMT; Nagao, 1984; Sumita et al., 1990), also called memory-based translation (Sato and Nagao, 1990), began as a data-driven MT paradigm separate from SMT and RBMT. In EBMT, translation fragments observed in parallel corpora are stored and used again later, allowing the translation process to make use of the largest fragments available. Researchers in the LFG framework applied this idea to their representation, leading to Data-Oriented Translation (DOT; Way, 1999; Hearne and Way, 2003), which stored tree fragments instead of words or phrases. In the major next step of this line of research, the principles of DOT were applied to LFGs deep syntax[2] via its f-structures (Graham et al., 2009; Graham, 2011). My second transfer pair extraction method, described in Section 6.2, is inspired by this work in LFG and also by Jellinghaus (2007).

3.2 Statistical and Neural Machine Translation

Rather than use hand-built rules, statistical machine translation (SMT) relies on automatically inferred word and phrase alignments. Och and Ney (2003) describe Giza++, a tool and iterative (i.e., expectation maximization, or EM) methodology for finding n-gram alignments. Lardilleux et al. (2012) offer an alternative, Anymalign, which does not iteratively optimize via EM, but rather finds alignments by randomly sampling "sub-corpora" of the training data. Anymalign focuses on low-frequency words, so by taking sub-corpora it makes high-frequency words into low-frequency ones within a sampling.

SMT is often phrase-based,[3] e.g., via the Moses toolkit (Koehn et al., 2007), but tree-based models are also popular. Yamada and Knight (2001) describe a model that parses the source sentence to a labeled syntax tree, then decodes to the target string. Chiang (2007) expands phrased-based SMT beyond contiguous words and phrases with local reordering

[2]A syntactic representation detached from word order, thus being close to a semantic representation.

[3]A *phrase* in SMT, unlike in syntactic frameworks like HPSG, is an unstructured n-gram of word tokens.

models to *hierarchical phrases*, which are empirically derived trees similar to syntax trees and make it easier to apply global reordering models. Chiang (2010) then translates with a source to target syntax (i.e., tree-to-tree) model. Gildea (2003) explored tree-to-string and tree-to-tree alignment and Eisner (2003) learned non-isomorphic tree mappings.

These tree based methods are relevant to my dissertation because I align semantic dependency structures which can be approximated as trees. Quirk et al. (2005), also Quirk and Menezes (2006), used syntactic dependency trees and combined the principles of SMT and EBMT to find **treelets** (i.e., the dependency-tree equivalent of a phrase). Aue et al. (2004) used semantic dependencies instead of syntactic dependencies and found transfer mappings for a graph-to-graph SMT system. A common way of mapping source to target structural information is through synchronous context-free grammars (SCFGs), which first appeared as syntax-directed transduction (Lewis II and Stearns, 1968), then as stochastic inversion transduction grammars (Wu, 1997), and later as SCFGs (Lopez, 2008), e.g., as implemented in the cdec (Dyer et al., 2010) and Joshua (Post et al., 2013) decoders. Ding and Arase (2014) used SFCGs to perform syntactic dependency-to-dependency translation for English and Japanese. Jones et al. (2012) use hyperedge replacement grammars to parse strings into AMR (see Section 2.12), then transform the representation into the target string, thus treating the AMR as an interlingua for translation. Using a meaning representation like AMR as an interlingua is not recommended,[4] but Xue et al. (2014) found that it was useful for narrowing the differences between the languages.

More recently, neural approaches to machine translation (NMT) have become a mainstream paradigm. The sequence-to-sequence models (Bahdanau et al., 2014) improve on SMT's token replacement and reordering strategies by reading in entire sentences (i.e., sequences), which are stored as an internal vector representation, then generating entire new sequences in the target language, which allows it to make greater use of context than SMT can. The addition of an attention mechanism (Luong et al., 2015) allows the system to focus

[4]See `https://github.com/amrisi/amr-guidelines`

on certain parts of the source sentence, which helps in long sentences and for translating names. NMT, however, suffers from out-of-vocabulary issues as the model's computational complexity increases significantly with larger vocabularies (Cho et al., 2014). There has been success in translating at the character level instead of the word level Lee et al. (2017), as the vocabulary of characters is much smaller than that of words. Furthermore, the character-level model is able to handle misspellings, neologisms, and code-switching into the target language (or other languages, if the model was trained multilingually), and it works without segmentation, which means it is not affected by segmentation errors by another tool. Others have used NMT for sequence-to-dependency mapping (Wu et al., 2017), which was shown to help in Chinese to English and Japanese to English tasks, and others have similarly found success with incorporating syntactic information into NMT (Aharoni and Goldberg, 2017; Chen et al., 2017). I do not use methods of NMT in this dissertation, but recent work on neural parsing to DMRS (Buys and Blunsom, 2017) and neural generation from DMRS (based on work by Konstas et al., 2017) inspire me to propose neural transfer as an area of future research (see Section 12.2).

Research in SMT is driven largely by its automatic translation quality evaluation metrics. Papineni et al. (2001) describe the BLEU metric, which is a measure of n-gram overlap between a translation and a reference sentence. The BLEU metric has widespread adoption within the machine translation literature, but it is known to have some deficiencies (see Section 9.1). Numerous variants of BLEU have been proposed, such as the NIST metric (Doddington, 2002), which gives more weight to certain n-grams and adjusts other penalties. The METEOR metric (Banerjee and Lavie, 2005; Lavie and Agarwal, 2007) was developed to be more robust to good kinds of variation (e.g., synonyms, minor word order differences, etc.). I use these three metrics for evaluation, as described in Section 9.1. I use BLEU because it is standard and METEOR because it is more adapted to the kinds of variation that my systems are likely to produce. I use NIST because it is not as strict as BLEU about conforming to a reference sentence but not as relaxed as METEOR.

3.3 Chapter Summary

There are multiple paradigms and methods of machine translation and even more techniques for fine tuning these methods to move the BLEU needle a little bit higher. I have described a small selection of rule-based, example-based, and statistical machine translation methods upon which I build my own methodology. Ultimately my implementation is built on top of the LOGON transfer machinery but my methods for extracting transfer rules are inspired by the Haugereid and Bond (2011, 2012) use of aligned predicate phrases, as well as observed dependency fragments, similar to what was done for LFG (Hearne and Way, 2003; Graham et al., 2009) and dependency treelets (Quirk et al., 2005; Quirk and Menezes, 2006).

My method typically requires the use of HPSG grammars to analyze and realize sentences. In theory the neural parsing system of Buys and Blunsom 2017 could be used to get DMRS representations from text, there is no check that the representations represent well-formed semantics, and may thus be incompatible with the transfer rules. More likely is that the robust generation techniques of Horvat et al. (2015); Horvat (2017) or Konstas et al. (2017) could be used to realize sentences even for malformed transfer outputs. I discuss this latter possibility in Section 12.2.

My transfer pair extraction methods may work with representations that are not DMRS, such as AMR or some other dependency representation, but these mappings would require a custom transfer solution, as the LOGON tools would be unable to accommodate a non-MRS representation. While other dependency representations encode similar relationships, their graphs are not structured like MRS representations, their edge and node labels are different, and they may make different decisions about how to handle complex phenomena including compounding, coordination, named entities, scopal modification, etc. As discussed in Chapter 2, the *MRS representations have many useful properties for transfer and translation, such as compositionality (which helps with consistency) and underspecification (which helps with data sparsity). In the next chapter I give an overview of my translation system and how a DMRS representation moves through the system to produce a translation.

Chapter 4
SYSTEM OVERVIEW

In this chapter I provide a general system description for semantics-based machine translation with automatically extracted transfer rules. I begin with an overview of the translation pipeline in Section 4.1. Section 4.2 gives an explanation of how I manage the information that traverses through the pipeline. Summaries of the parsing, transfer, and generation processes follow in Sections 4.3 to 4.5. Then I explain how candidate selection and evaluation fits in the pipeline in Section 4.6.

4.1 Translation Pipeline

The translation pipeline takes a textual sentence as input and produce a textual sentence as output. In order to get from the source sentence to the target sentence, the source sentence is first analyzed, yielding zero or more source-language semantic representations. Each source-language semantic representation is then transferred to zero or more target-language semantic representations. Each target-language semantic representations is used to generate zero or more candidate target-language sentences. A path from the source sentence to a single parse, transfer, and realization is called a **hypothesis**, and the collection of all paths from a single source sentence is a **hypothesis set**. In the last step, a single translation is selected from the candidate realizations for each hypothesis set. A simplified illustration of the translation process is given in Fig. 4.1.

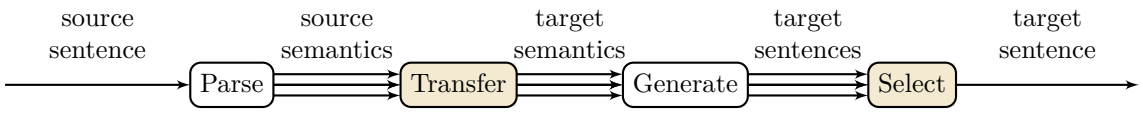


Figure 4.1: Simplified translation pipeline

I use existing tools and resources for the parsing and generation steps. The HPSG grammar-based processor ACE[1] is the primary tool for transforming language representations, and is the central process for the parsing, transfer, and generation steps. For parsing and generation I use existing grammars, but I customize the transfer grammars used in my experiments. The selection step is implemented by me. Further explanation about the internal components of these steps and the data flow between them will be given in this chapter.

Each stage of the translation process can yield multiple results (p parses, x transfers, and r realizations), so if no limits are placed on their outputs the search space quickly becomes unmanageable. For example, if I simply took the first 5 parses, transfers, and realizations, there would be up to $pxr = (5)(5)(5) = 125$ possible translations for each input sentence. Furthermore, transfer and realization ranking models are relatively underpowered compared to parse ranking models, so it is more likely for the preferred parse to appear in the top p results than for the preferred transfer in the top x or preferred realization in the top r, meaning I can potentially get better results by increasing the number of results for an item at the cost of computing time. While each component is efficient at enumerating packed results (Carroll and Oepen, 2005; Zhang et al., 2010) for a given input, it is still costly to process many inputs. That is, it would be faster to enumerate 50 results from a single input than 5 results from 10 inputs because the former only needs to perform chart parsing to construct the packed forest once and then enumerate the results, while the latter performs chart parsing 10 times. In terms of my translation pipeline, increasing the number of parses is the most costly, because it increases the number of inputs to process for transfer and, indirectly, realization. Therefore in my experiments I keep parses and transfers restricted to 5 each, but I allow up to 20 realizations, as illustrated in Fig. 4.2.

Velldal (2008) built the realization rerankers I use in my pipeline, which is a discriminative log-linear model over derivation tree paths, similar to the established approach for parse

[1] `http://sweaglesw.org/linguistics/ace/`

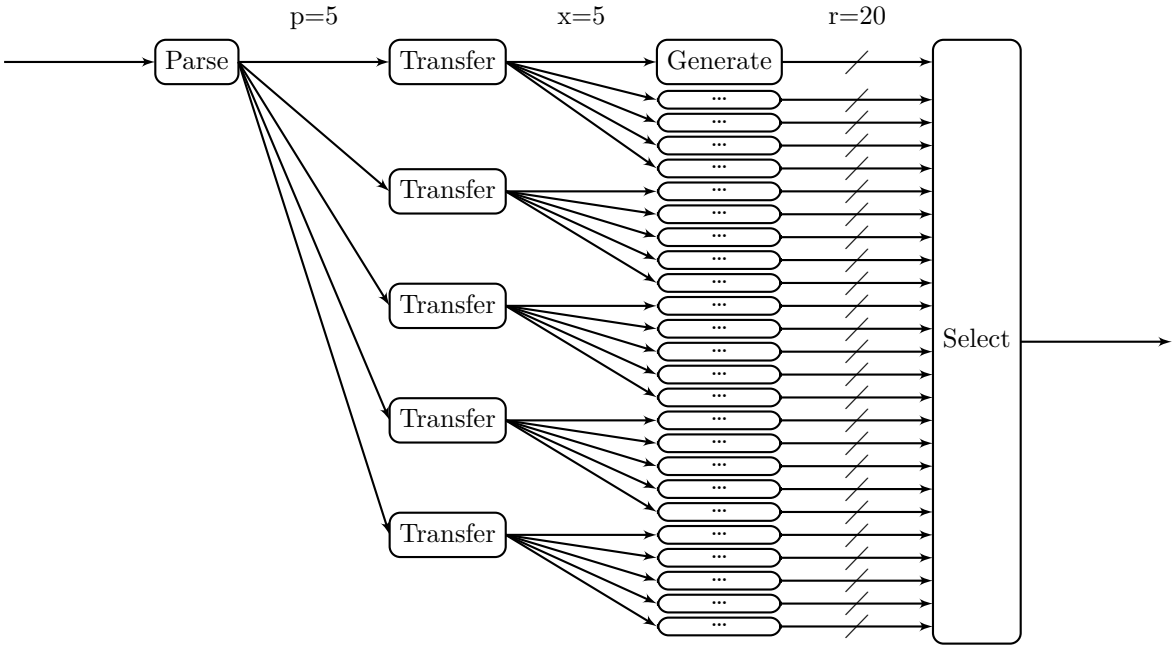


Figure 4.2: Pipeline fan-out

reranking (Toutanova et al., 2005). Velldal also built a generative n-gram model over realized tokens into the realization reranker, a simpler transfer reranker that trains an n-gram model of EDS triples (Oepen and Lønning, 2006), and an end-to-end translation reranker that uses a log-linear model taking the scores of the previous three rerankers as features, but I do not make use of these models. ACE only fully utilizes the existing parse reranking model.[2] For realization ranking, ACE uses the discriminative model, but not the n-gram model, and there is no reranking of transfer outputs. The goal of my research is not to create a system with high-quality top-1 outputs, but to increase the coverage of the transfer grammar and to increase its capability to produce high-quality outputs. The lack of the transfer reranker, n-gram realization reranker, and end-to-end reranker is therefore not a big issue, as I can examine all the translation outputs for an input item rather than only the top 1. Section 4.6

[2]ACE `0.9.26`, which I use, unlike the LKB (Copestake, 2002) only incorporates the discriminant model for parsing and generation, and not the n-gram model for generation or MRS-triple model for transfer. Future versions of ACE may include these models.

describes how I evaluate the quality when considering all outputs.

4.2 Pipeline Information Management

At the simplest conceptual level, a source sentence is fed to the parser, the relevant data is fed through to each component, and a target sentence is produced, but there is more information being passed around than is described by this view. Each abstract component (parser, transferer, generator, and selector) takes a data structure containing the relevant representation along with metadata describing the representation and where it came from. These data structures are implemented as a variant schema of [incr tsdb()] (Oepen and Flickinger, 1998) tables, allowing the results of the full translation pipeline to be stored in one profile (i.e., database). See Appendix B for a description of these structures.

Bond et al. (2011) used the [incr tsdb()] grammar profiling software (Oepen and Flickinger, 1998) to manage similar kinds of information in their machine translation pipeline. The standard [incr tsdb()] schema does not allow a single profile to be used for the whole machine translation pipeline, as it only has one table for storing system outputs. It also contains many tables useful for grammar profiling, but irrelevant for machine translation. Bond et al. circumvented the single-profile limitation by storing the profiles generated by each component as nested directories. That is, they had one profile for parse results, then separate transfer profiles for each hypothesis (i.e., one containing the first parse result for each item, another for the second parse result, etc.), and a further level of similar nesting for realization results for each transfer profile. I prefer to keep one profile for the whole pipeline with separate tables for the outputs of each component, and with none of the irrelevant tables from the standard schema. Rather than embed result identifiers from earlier tasks into directory names, I store them explicitly as columns in the tables. This approach is functionally equivalent to that of Bond et al., but I find it less cumbersome to work with and more efficient with disk space.

One drawback of my approach is that the [incr tsdb()] software does not work with non-standard schemas, and ACE does not, by itself, maintain metadata during processing. There-

fore I wrap the components with a modulator-demodulator mechanism for pairing processor outputs with the corresponding metadata from the inputs. For example, the modulator wrapping the parser extracts the source sentence (`i-input`) field from an input item and feeds it to the parser. The demodulator collects all the results for that sentence and assigns the original item's identifier to each result's item id (`i-id`) field, since the parser does not keep track of that information itself. This approach is also used in the transferer, copying both the item id and the parse id, as well as the generator, copying the item, parse, and transfer ids. Thus, each translation hypothesis contains the information needed to recover its original item, parse result, and transfer result.

4.3 Parsing

Input sentences are parsed using one of the existing broad-coverage HPSG grammars with the ACE processor, yielding zero or more semantic representations per input sentence. I use the English Resource Grammar (ERG; Flickinger, 2000) for English sentences and the Jacy grammar (Siegel et al., 2016) for Japanese. Note that both the ERG and Jacy are used for parsing when I build a transfer model (summarized below in Section 4.4), but at run-time the translation pipeline parses with Jacy and generates with the ERG.

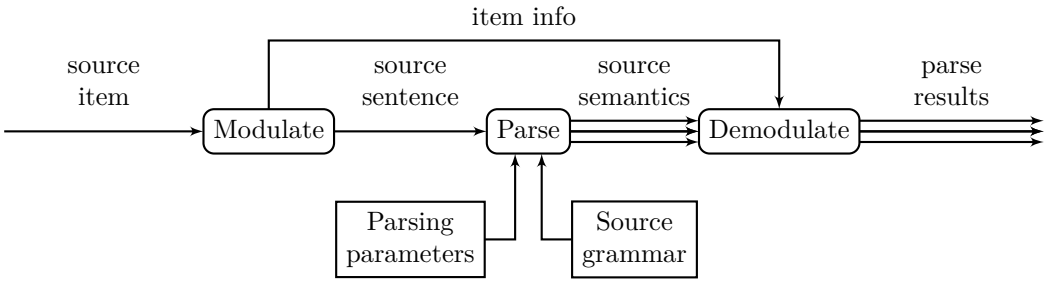


Figure 4.3: Expanded view of the parsing component

Fig. 4.3 illustrates the data flow of the parsing component. In my experiments, the *parse* process is ACE and the *source grammar* is Jacy. The *parsing parameters* include constraints such as the number of results (p) to allow, memory or time limits on processing, whether

the input is pre-tokenized, etc. See Chapter 9 for information about the parameters used in my experiments. For each source item passed to the modulator, the sentence is sent to the parser, and between zero and p semantic representations are produced. The demodulator takes the information from the input item and pairs it with each semantic output, resulting in parse results that are sent to the next step.

The core of the parser (i.e., *parse*, *parsing parameters*, and *source grammar*) utilizes a number of subsystems for processing each input sentence. The sentence may be pre-tokenized (e.g., Japanese data for Jacy is tokenized using MeCab morphological analyzer (Kudo, 2005)), or it will be tokenized with grammar-defined rules (using REPP (Dridan and Oepen, 2012)) to separate individual lexical elements from each other and from punctuation, and to remove extra-linguistic material like HTML markup. If the tokens can be matched with lexical entries in the grammar, the token-mapping stage (Adolphs et al., 2008) takes those tokens and produces feature structures that the grammar can use for parsing. The grammar rules are then applied via a chart-style parsing algorithm with unification of feature structures. For each input sentence, many parses may be found, and a parse-selection model ranks them. The semantic representations constructed during parsing are specific to the grammar and grammar version that produced them, so the representations are normalized on output to a grammar-external variant via a variable-property-mapping (VPM). The VPM can change variable names (e.g., `event5` → `e5`), and can add, remove, or modify variable properties, such as those for tense/aspect or person/number/gender.

4.4 Transfer

The transfer component (Lønning et al., 2004) takes a semantic representation as input and produces zero or more target semantic representations. Transfer is a function built into both ACE and the LKB, and while it utilizes many structures used in monolingual grammars (e.g., typed feature structures, VPMs), it is not a unification-based process like with parsing and generation, but rather a graph-rewrite process. The purpose of transfer is thus the transformation of semantic representations. Generally the source and target representations

are MRSs defined by grammars for different languages (i.e., transfer for translation), but it is also possible to target the same language or grammar as the input (e.g., for summarization), or to use source or target MRSs that do not come from grammars (e.g., from a string-to-graph parser (Buys and Blunsom, 2017) or to a graph-to-string generator (Horvat et al., 2015)). This dissertation is only concerned with transfer for translation.

A transfer grammar encodes the mapping between representations, and it is a separate grammar from those corresponding to the source or target representations. Unlike monolingual grammars, a transfer grammar is not reversible;[3] in order to flip the source→target direction, a separate transfer grammar is required. Transfer grammars can be much larger than monolingual grammars, as they cover the semantics defined by two different monolingual grammars. I use and build on the JaEn transfer grammar (Bond et al., 2005, 2011) for translating from Japanese (as analyzed by Jacy) to English (as analyzed by the ERG).

The core of a transfer grammar defines rule types for MRS transfer rules (MTRs),[4] a small number of hand-written MTR instances and possibly some automatically generated MTRs (e.g., enumerated number names), and also other technical particulars, such as the variable-property mappings tailored to the source and target grammars and the order of inclusion of files containing MTRs. The hand-written MTRs include monolingual fix-up or normalizing rules, which prepare the source-representation outputs or clean up the transfer outputs in order to streamline the transfer process. The hand-written MTRs also include some bilingual instances that capture correspondences that are difficult to acquire automatically. Figs. 4.4 and 4.5 show two hand-written MTRs: the first normalizes the predicates for the kanji and hiragana orthographic variants 分かる *wakaru* and わかる *wakaru*, both meaning "to understand", and the second captures the idiomatic translation 嘘をつく *uso-wo tsuku* "tell a lie". The second is one that my methodology could extract automatically, while the first is not. MTR application is ordered, so the order of MTR file inclusion is important for translation performance. This definition includes two placeholder files which will be

[3]Parts of transfer are reversible, but as a whole a transfer system is not reversible.

[4]The components of MTRs are described in Section 7.1.1.

filled by the single and multi-word expression (MWE) MTRs, respectively, that I create (see Chapter 9). Note that this ordering is separate from the automatic rule ordering I perform, which is the order of rules within a file, as described in Section 7.2.2.

```
wakaru_v_1--wakaru_v_3_jf := arg12_v_mtr &
[ INPUT.RELS < [ PRED "ja:_wakaru_v_1_rel" ] >,
  OUTPUT.RELS < [ PRED "ja:_wakaru_v_3_rel" ] > ].
```

Figure 4.4: Monolingual MTR in JaEn for normalizing alternate orthographies

```
tsuku-lie := arg12_v_mtr &
[ CONTEXT.RELS < [ PRED "~^ja:_uso_n", ARG0 #x  ] >,
  INPUT.RELS   < [ PRED "ja:_tsuku_v_6_rel", ARG0 #2 ] >,
  OUTPUT.RELS   < [ PRED "_tell_v_1_rel" ] > ].
```

Figure 4.5: Hand-written MTR for the idiomatic 嘘をつく *uso-wo tsuku* “tell a lie [lit: breathe a lie]”

The core transfer grammar is augmented with automatically extracted MTRs which make up the majority of the rule instances. This augmented, or customized, transfer grammar has rules selected for a particular set of inputs or for a genre. In the methodology employed by the JaEn transfer grammar, the rule instances in the extended grammar are automatically extracted from bilingual corpora (Bond et al., 2011). Previous efforts relied on matching semantic templates to predicates found by a bilingual n-gram aligner (Haugereid and Bond, 2011, 2012). I also use an n-gram aligner to find translationally equivalent predicates but, in contrast, I extract the rules without the use of templates. I also align semantic subgraphs directly, i.e., without the n-gram aligner outputs, as a second method. Extracted transfer pairs are kept in a transfer pair store, and a subset of this store is selected for a transfer task. The selection of the subset is partially for performance reasons—the larger the grammar, the

more memory for storage and time for processing it requires—and partially because I want to filter out bad transfer pairs that introduce noise into the hypothesis set.

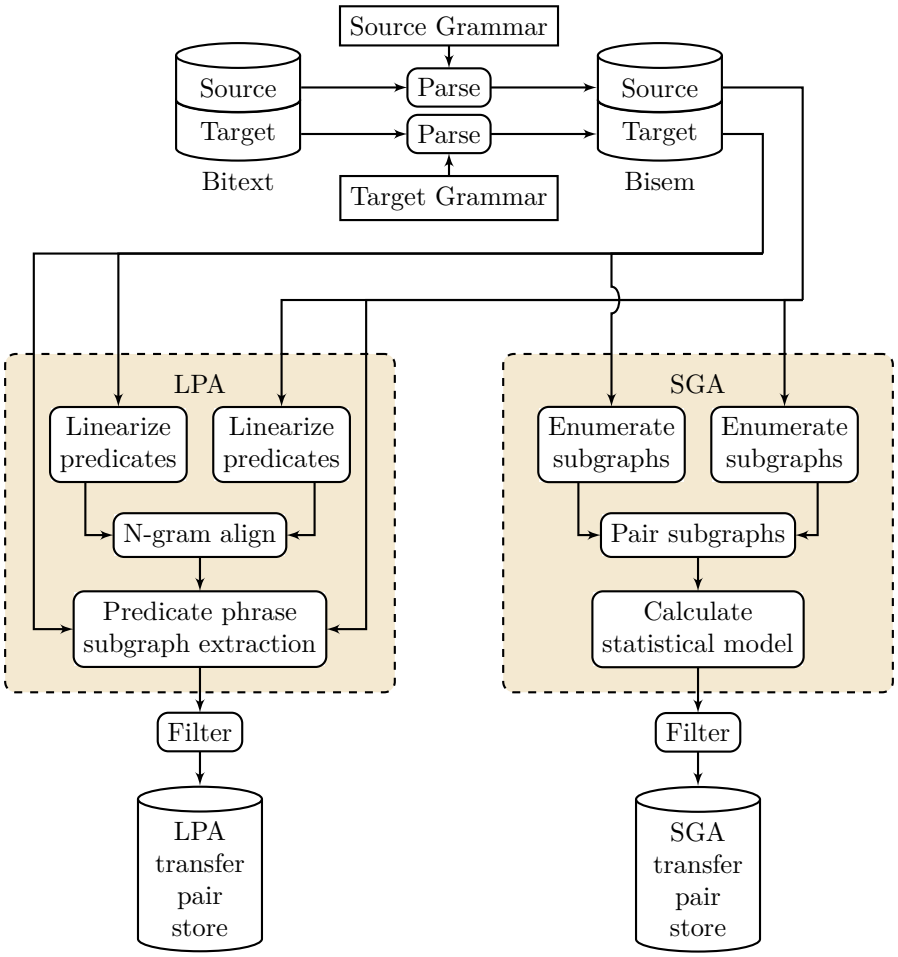


Figure 4.6: Building the transfer pair store

The process of building transfer pair stores for the two experimental systems (see Chapter 6) is illustrated in Fig. 4.6. The source and target semantic representations in the bilingual semantic (**bisem**) corpus (the result of parsing the bitext corpus with the source and target grammars) are fed to both the Linearized Predicate Alignment (LPA) and Subgraph Alignment (SGA) transfer-pair extractors. LPA (see Section 6.1) linearizes the predicates, bilingually aligns n-grams of predicates, then uses those alignments with the original seman-

tic representations to extract bilingual semantic subgraphs. SGA (see Section 6.2) is roughly the inverse of LPA. First, unaligned subgraphs are extracted from the semantic representations, then the source subgraphs are paired with every compatible target subgraph within a bisem pair, and finally the paired subgraphs are counted in order to build a statistical model over the pairs. The output of both systems—bilingually paired semantic subgraphs—are then subjected to the same kinds of filters (see Section 6.3), although with different parameters, and put into their respective transfer pair stores. During a transfer task, a subset of task-relevant pairs will be selected from the stores, converted into the transfer rule format (see Section 7.3), and compiled into a customized transfer grammar.

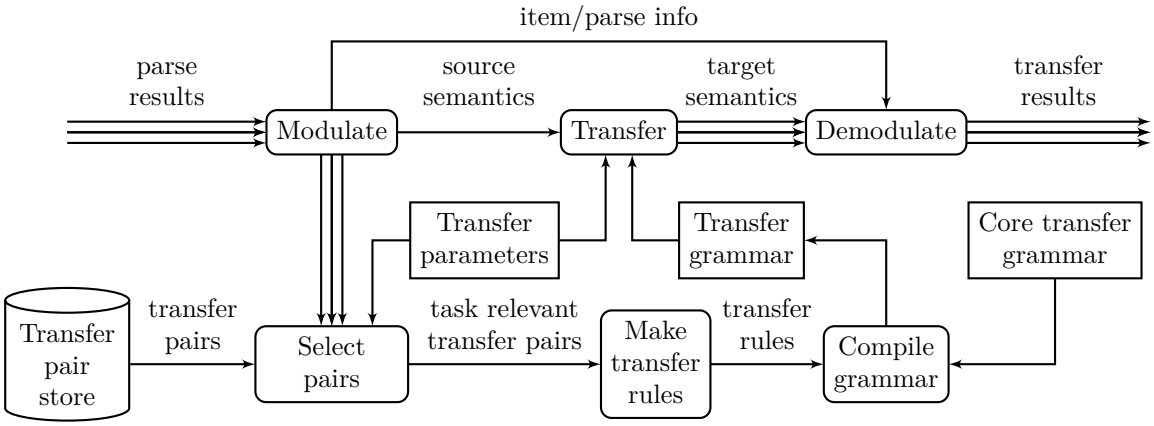


Figure 4.7: Expanded view of the transfer component

Fig. 4.7 illustrates the data flow of the transfer component. The main processes in this component mirror those for parsing, but I add two more processes in order to enable experimentation on transfer grammars. The *select pairs* process selects a subset of the extracted transfer pairs (from the *transfer pair store*) for use in the transfer grammar. *Transfer parameters* inform the selection about constraints relevant to a current experiment, such as filters on the size or shape of semantic subgraphs (see Section 6.3). Also, all incoming source semantic representations are inspected so only relevant transfer pairs are selected (see Section 7.2), which keeps the transfer grammar to a manageable size. The *make transfer rules*

process then converts the subgraph pairs into an MRS transfer rule, as described in Section 7.3. The *compile grammar* process then uses these transfer rules with the core JaEn grammar to create the transfer grammar used in the *transfer* process.

4.5 Generation

Input semantic representations to the generator produce zero or more realized sentences. For the translation pipeline, only the ERG is used for generation, as English is the target language in my experiments. A VPM maps variable properties to the form required by the target grammar. Fig. 4.8 illustrates the data flow of the generation component.

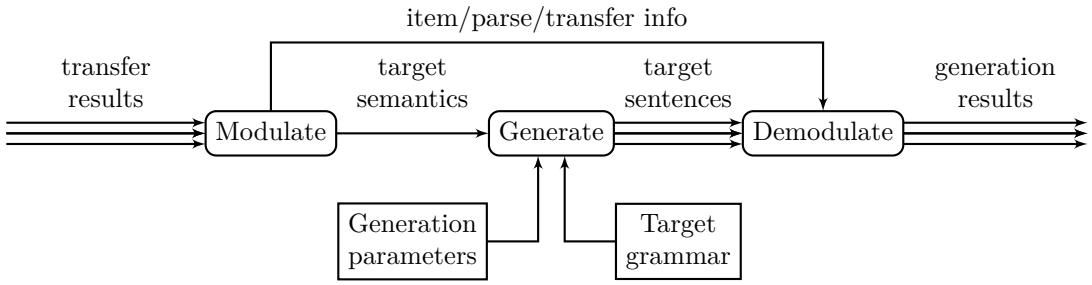


Figure 4.8: Expanded view of the generation component

4.6 Translation Selection

The generator produces r realizations for the x transfers and p parses of an item, and with the constraints on results I use in my experiment, this could yield potentially $(5)(5)(20) = 500$ realizations in a hypothesis set, so I implement two methods for selecting a representative realization from each set, following the lead of Oepen et al. 2007. The first method, **First**,[5] selects the first available realization within a hypothesis set, while the second method, **Oracle**, selects the hypothesis that maximizes the BLEU score for the item. The data flow for the First and Oracle methods are shown in Figs. 4.9 and 4.10.

[5]Oepen et al. (2007) call this **first translation**.

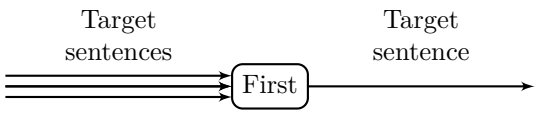


Figure 4.9: Expanded view of the selection component: First method

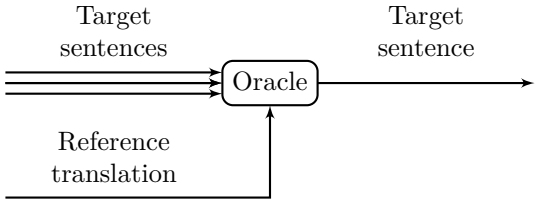


Figure 4.10: Expanded view of the selection component: Oracle method

The First method selects the first available realization from the group of realizations with the same `item-id`. This method effectively selects the highest ranked parse resulting in a realization, the highest ranked transfer resulting in a realization, and the highest ranked realization, each according to their respective ranking models, if any. Fig. 4.11 shows an example of first-selection. In the example, the first parse (id `0`) has two transfers but neither of them have any realizations, and the second parse (id `1`) has no transfers, so these are both skipped. The third parse (id `2`) has two transfers, both with realizations, so the first transfer and its first realization are selected.

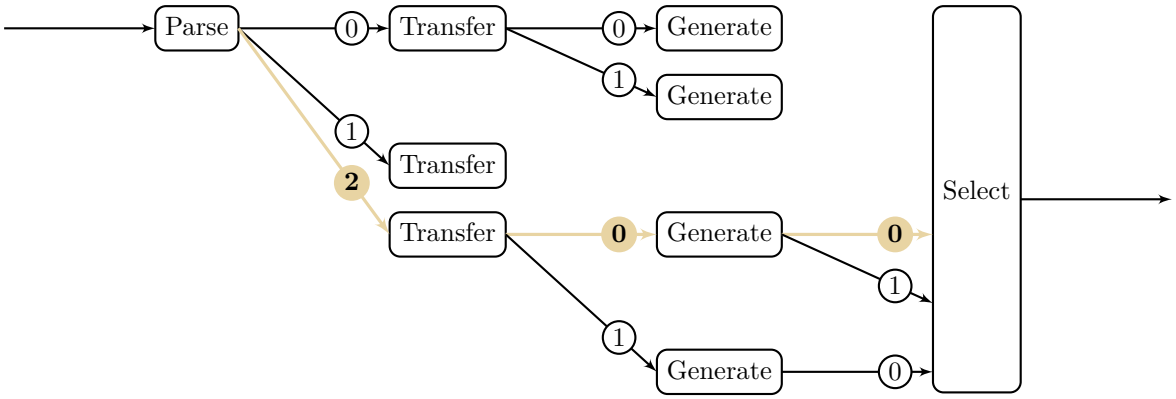


Figure 4.11: First-selection of translation hypotheses

In order to find the realization that maximizes the BLEU score, the Oracle method accumulates all realizations for an item, then calculates a smoothed BLEU score[6] for each one compared to the reference translation. The realization, and thus its hypothesis path, with the highest score is then selected, regardless of the order or reranking scores for the hypotheses' parses, transfers, and realizations. This method can only be performed for evaluation, as an online system would not have access to reference translations. The goal of this method is to show the capability of the system to produce high-quality translations, i.e., the upper bound of its performance. With better reranking on the pipeline components and for end-to-end translation, the translation selected by the First method could become the one selected by the Oracle method as well.

4.7 Chapter Summary

This chapter has described the overall architecture of the machine translation pipeline I utilize in my experiments. The primary stages of parsing, transfer, and realization expand the search space of translations available to my systems, and the selection stage narrows that search to a single translation for each input item. While I'm using existing tools and resources, namely the ACE processor, the Jacy and ERG monolingual grammars, and the core JaEn transfer grammar, I have built my translation pipeline around these tools and resources so that I can track translation hypotheses throughout the process. The pipeline code and accompanying scripts for data preparation are freely available as the XMT package at `https://github.com/goodmami/xmt`. The knowledge source of the transfer stage, the transfer pair store, is built using information from the parsing stage, using bilingual inputs. Later chapters describe how I extract transfer rules to augment the transfer grammar (Chapters 5 to 7) and design experiments (Chapters 8 and 9) in order to evaluate the methods (Chapters 10 and 11).

[6]BLEU scores are meaningful when computed for a large corpus, but less so when comparing individual items. The smoothed score helps in this regard. See Section 9.1 for more information.

Chapter 5
SEMANTIC OPERATIONS

The automatic extraction of transfer rules involves the manipulation, inspection, and tabulation of semantic representations, so in this chapter I cover a number of the semantic operations I use in my extraction methods. Semantic transfer as implemented by the LOGON transfer machinery (Lønning et al., 2004) is a resource-sensitive rewrite process on MRS structures (see Chapter 7 for more information), but much of my workflow operates on DMRS (Copestake, 2009) data, so in Section 5.1 I describe the conversion from MRS to DMRS, and in Section 5.2 I describe the conversion from DMRS back to MRS. I also describe various ways of inspecting DMRS structures, from semantic tests in Section 5.3 to traversals in Section 5.4. The standard way of serializing DMRS is in an XML format, which is neither compact nor an intuitive representation for semantic dependencies, so in Section 5.5 I define a procedure to serialize DMRS into PENMAN notation, as well as a normalization procedure to make equivalent PENMAN-encoded DMRS structures string-comparable. Directly relevant to the automatic creation of transfer rules from corpora is the extraction of semantic subgraphs, described in Section 5.6, and the simplification of DMRS representations in Section 5.7.

5.1 MRS to DMRS Conversion

In Section 2.6 I explained how MRS is an expressive meta-level language for encoding, e.g., logical form, and in Section 2.10 I described DMRS as a dependency representation that captures nearly all of the information encoded in MRS. One principle difference is that MRS captures quantifier scope through label sharing, where DMRS encodes label equality between two nodes on the edge connecting them, thus full scopal conjunctions (e.g., for more than two nodes) can be recovered by traversing the graph to find transitive equalities. The process for

converting from MRS to DMRS was first described in theoretical terms in Copestake 2009 with a Lisp implementation in the LKB (Copestake, 2002). Here I give a more concrete description of the conversion process based on my Python implementation in PyDelphin.

5.1.1 Converting Elementary Predications to Nodes

The first and simplest procedure is the conversion of Elementary Predications (EPs) into nodes, as shown in Algorithm 1. The procedure iterates through all the EPs in the MRS instance and creates exactly one node for every EP. EPs in the MRS formalism do not have unique identifiers so one is assigned to each new node.[1] The EP's predicate is copied over directly to the node. As DMRS does not have variables, any variable properties in MRS for some variable v are instead assigned to the node whose corresponding EP has the intrinsic variable v. DMRS properties[2] contain an additional property `cvarsort` that stores the type of the original variable.[3] DMRS nodes do not contain arguments as EPs do, but they do contain the values of any constant argument, which are pseudo-arguments in MRS. So for the final conversion step the value of the EP's CARG role is assigned as the node's *carg* attribute.[4]

---

[1]Conventionally node identifiers are integers starting from 10,000 but I start from 1 in Algorithm 1.

[2]In DMRS, properties are called **sortinfo**. I call them **properties** here.

[3]This information is useful in some specific cases, such as color names. In the ERG's analysis of colors the intrinsic variable of the color's EP will be an `e` (eventuality) when used as an adjective (e.g., *She ate the red apple* or *The apple is red*) but will be an `x` (instance) when the color is used as a noun (e.g., *Red is her favorite color*). Some could argue that this information is recoverable from an analysis of the semantic graph; e.g., when the color is quantified over, it's EP's ARG1 role will be unfilled.

[4]In Algorithm 1 I hard-code the constant-argument role CARG. While this role name is customizable in some grammar processors, such as the LKB, no grammar to my knowledge has ever used a different role name. PyDelphin also allows the name to be customized, defaulting to CARG. In DMRS, the *carg* attribute is defined as part of the schema, which is available at `http://svn.emmtee.net/trunk/lingo/lkb/src/rmrs/dmrs.dtd`.

Algorithm 1 Converting MRS EPs to DMRS Nodes

Input: R is a list of Elementary Predications as $\langle$*pred*, *label*, *var*, *arg*$\rangle$ tuples
Output: N is a list of Nodes as $\langle$*id*, *pred*, *prop*, *carg*$\rangle$ tuples, where $|N| = |R|$ and N_i corresponds to R_i

function NODES(R)
  for $i \leftarrow 1, |R|$ **do**
    $pred \leftarrow R_i^{pred}$
    $Prop \leftarrow$ VARPROPS(R_i^{var}) $\cup$ {("cvarsort", VARSORT(R_i^{var}))}
    $carg \leftarrow R_i^{Arg}$("CARG") ▷ Constant argument may be nil
    $N_i \leftarrow \langle i, pred, Prop, carg \rangle$

5.1.2 Representative Node Selection

The conversion of MRS argument structure and quantifier scope to DMRS links is more complicated than the conversion of EPs to nodes and one contributor to that complexity is representative node selection. Representative node selection is described in Sections 2.8 and 2.10 as that which allows EDS (Oepen et al., 2002; Oepen and Lønning, 2006) and DMRS to encode the scopal arguments of MRS as regular graph edges. Due to its importance in MRS to DMRS conversion and its relative complexity, I describe the process here separate from the creation of links, which is covered in Section 5.1.3. Algorithm 2 defines the process for finding representative nodes, although it is implemented as finding the indices of EPs that correspond to representative nodes.

Algorithm 2 Finding the indices of representative EPs

Input: R is a list of Elementary Predications as $\langle$*pred*, *label*, *var*, *Arg*$\rangle$ tuples
Input: h is the label shared by some non-empty set of EPs in R
Output: I is a list of indices of representative EPs in R, sorted by representativeness

function REPRESENTATIVEEPINDICES(R, h)
  $Conj \leftarrow \{i \in \{1, \ldots, |R|\} : R_i^{label} = h\}$ ▷ Indices of EPs sharing label h
  $Vars \leftarrow \{R_i^{var} : i \in Conj\}$ ▷ Intrinsic variables in *Conj*
  $Reps \leftarrow \{i \in Conj : (\nexists\ role)[R_i^{Arg}(role) \in Vars]\}$
  $I \leftarrow$ REPSORT($Reps, Conj, R$)

For a list of EPs R and a label h, I first find the indices of the subset of EPs that share the label (the EP conjunction) and assign them to the variable *Conj* (line 2). For notational convenience, I also assign the set of intrinsic variables for this EP conjunction to the variable *Vars*, which is used in the following line. The indices of representative EPs are then defined (line 4) as the subset of *Conj* where the EP at index i (R_i) has no outgoing arguments to other EPs in the conjunction (i.e., EPs with a role argument value that is in *Vars*). This subset of *Conj*, assigned to *Reps*, will have at least one index, but it may have more than one. When there is more than one index in *Reps*, however, not all correspond intuitively to the **most** representative node. Therefore, in line 5, I call RepSort() on the set of *Reps*, which prefers EPs that are either quantifiers or quantifiees,[5] then those with the most incoming edges in the conjunction (the one(s) most often modified), and any remaining ties are resolved by preferring lower values of the index itself (i.e., those that come first in the original order). The sort thus returns a list ordered by representativeness which allows me to select a single most representative EP. The purpose of returning a sorted list rather than *Reps* will become clear after the following example.

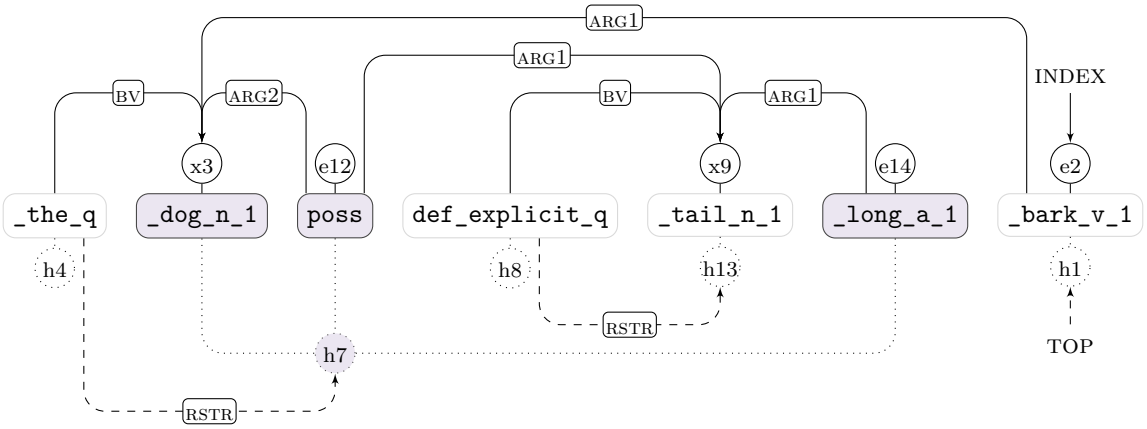


Figure 5.1: MRS for *The dog whose tail is long barked.*

[5]These are in fact separate cases, as a quantifier and its quantifiee should never be in a conjunction. When a quantifier is in a conjunction, it is likely an instance of **predicate modification** where the quantifier itself is modified, as in *nearly every dog barks*. When a quantifiee is in a conjunction, it is likely a nominal entity being modified, as the EP for *dog* is in Fig. 5.1.

Consider the MRS, as analyzed by the ERG, for *the dog whose tail is long barked* in Fig. 5.1.[6] The label h7 captures an EP conjunction of three EPs—`_dog_n_1`, `poss`, and `_long_a_1`—but there is no explicit argument of `_long_a_1` that selects `_dog_n_1`, nor transitively via `poss`, so DMRS argument links cannot capture the full scopal conjunction. There are therefore two representative EPs, `_dog_n_1` and `_long_a_1`, and two choices for inserting a MOD/EQ edge between them: one from `_long_a_1` to `_dog_n_1`, as shown in Fig. 5.2; and one from `_dog_n_1` to `_long_a_1`, as shown in Fig. 5.3. The first option, where `_dog_n_1` is being modified, is the more intuitive one, as *dog* is primary entity under discussion in *the dog whose tail is long*, and it is selected by the REPSORT() function as `_dog_n_1` is the only one being quantified.

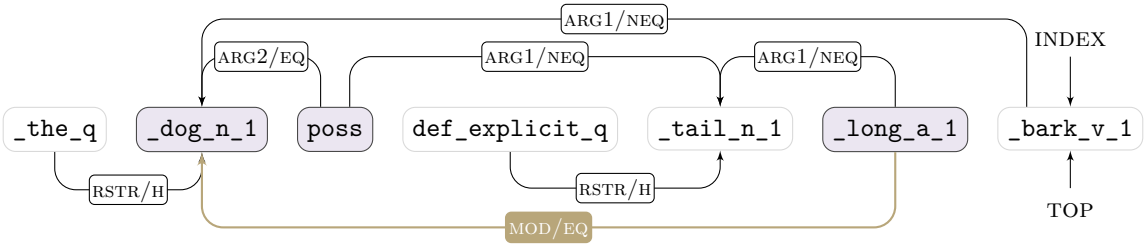


Figure 5.2: DMRS for *The dog whose tail is long barked.*

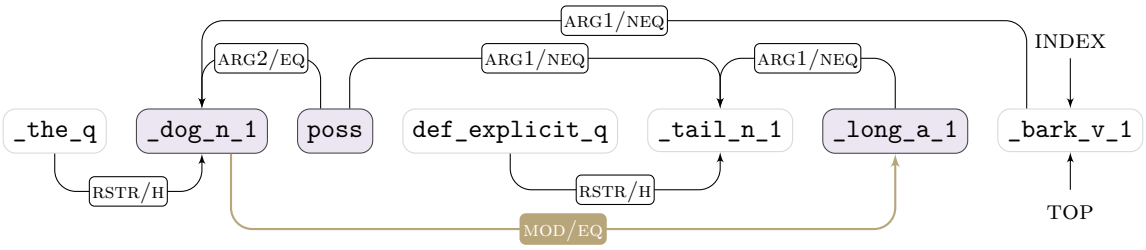


Figure 5.3: Alternative DMRS for *The dog whose tail is long barked.*

[6]The example is a slightly simpler version of the one given in Copestake 2009: *the dog whose toy the cat bit barked.*

5.1.3 Converting Elementary Predications and Handle Constraints to Links

With the definition of representative node selection in Section 5.1.2, I can now define link creation. The procedure, shown in Algorithm 3, has two main sections: the first creates DMRS links that correspond to MRS arguments (called **argument links**), with each link's target (*to*) and post-slash label (*post*) determined by the nature of the MRS argument; and the second part finds any necessary links to capture relationships not covered by the argument-derived links (**non-argument links**).

The argument links are created first (lines 3–23), from MRS arguments. Of these, links for the restriction (RSTR role) of quantifier EPs are handled first, separate from the other argument links.[7] For all other EPs, I iterate over each role argument and create the appropriate links for non-scopal arguments, label-selecting arguments, and hole-selecting (qeq) arguments. For non-scopal arguments, the target EP is found via its intrinsic variable, which is the value of the argument. If the target EP shares a label with the source EP, the link gets a *post* of `EQ`, otherwise it is `NEQ`. For label-selecting arguments, the target is the first representative EP for the label, and the *post* is `HEQ`. Finally, for hole-selecting arguments, the hole is qeq the label of an EP conjunction containing the target, which is then selected as the first representative EP for the label. The *post* for hole-selecting arguments is `H`.

Non-argument links are created last (lines 24–30) from MRS label equalities not evidenced by argument links. I find these by looking for representative EP lists with more than one element, and take the first in the list as the most-representative. All other EPs in the list of representative EPs are dependents, and I create a link with a special `MOD` value for the *role* and `EQ` for *post*.

5.2 DMRS to MRS Conversion

Converting to DMRS from MRS is not as straightforward as MRS to DMRS conversion, as unpacking and reassembling the information in the nodes and links to EPs and qeqs involves

[7] For quantifiers, only the restriction argument is made into a link for reasons discussed in Copestake 2009.

Algorithm 3 Converting MRS EPs and qeqs to DMRS Links

Input: R is a list of Elementary Predications as $\langle pred, label, var, Arg \rangle$ tuples
Input: $Q : lhs \rightarrow rhs$ is a function mapping a hole lhs to the label rhs it is qeq
Output: L is a list of Links as $\langle from, to, role, post \rangle$ tuples

```
1  function LINKS(R, Q)
2      N ← NODES(R)
3      for i←1, |R| do                                          ▷ Argument links
4          from ← N_i^id
5          if R_i is a quantifier then
6              to ← N_j^id, where R_j^var = R_i^var, i ≠ j      ▷ j is the index of the quantified EP
7              append ⟨from, to, "RSTR", "H"⟩ to L
8          else
9              for all ⟨role, val⟩ ∈ R_i^Arg do
10                 if val is the intrinsic variable of some EP in R then
11                     if R_i^label = R_j^label, where R_j^var = val then   ▷ Do they share a label?
12                         post ← "EQ"
13                     else
14                         post ← "NEQ"
15                 else if val is the label of some EP in R then
16                     j ← first index in REPRESENTATIVEEPINDICES(R, val)
17                     post ← "HEQ"
18                 else if val is a hole in Q then
19                     rhs ← Q(R_i^Arg(role))
20                     j ← first index in REPRESENTATIVEEPINDICES(R, rhs)
21                     post ← "H"
22                 to ← N_j^id
23                 append ⟨from, to, role, post⟩ to L
24     for all h ∈ {r^label : r ∈ R} do                         ▷ Non-argument links
25         I ← REPRESENTATIVEEPINDICES(R, h)
26         if |I| ≥ 2 then
27             from ← N_i^id, where i ← I_1
28             for j←2, |I| do
29                 to ← N_i^id, where i = I_j
30                 append ⟨from, to, "MOD", "EQ"⟩ to L
```

several partial steps. The algorithm for this process is given in Algorithm 4, and the steps are as follows: step 1 (lines 2–5) pre-assigns scope labels to node ids; step 2 (lines 6–12) initializes the EPs with information from the nodes; and step 3 (lines 13–25) assigns EP arguments and creates qeqs from information on the links. The algorithm uses two functions that I will only describe here: ConnectedEqComponents(L) returns a list of node id sets where each set is a connected component of a graph where the vertices are the nodeids and the edges are links with a *post* value of `EQ`;[8] and NewVar(t[,P]) is a stateful function that returns a new variable with type t and the next available variable id (an integer), and optionally assigns the properties P to the variable.[9]

Step 1 builds a mapping H of node ids to labels (scope handles). In an MRS, multiple EPs may share a label (called an **EP conjunction**), but at this point in the algorithm there are no EPs, and the scope information is not present on the nodes. Therefore this step discovers which subsets of nodes correspond to (eventual) EP conjunctions via the ConnectedEqComponents() function, and instantiates a new label via NewVar() for each subset and assigns all node ids in the subset to that label.

Step 2 instantiates the EPs from their corresponding nodes and places all information available on the node onto the new EP. The predicate is assigned directly, and the label is taken from the mapping H created in step 1. All EPs except for quantifiers get a unique intrinsic variable. Quantifiers do not get intrinsic variables as they get a **bound variable** instead, which is the intrinsic variable of the EP they quantify over. For this reason, quantifiers' bound variables are set in step 3, after all other EPs have gotten an intrinsic variable. The intrinsic variables of non-quantifier EPs are created via the NewVar() function using the variable type given in the node's `cvarsort` property, and all other properties are assigned to the variable. Step 2 also sets the value of any constant arguments if present on the corresponding nodes.

---

[8] For example, given links $\langle 0, 1, \texttt{ARG1}, \texttt{EQ}\rangle$, $\langle 2, 1, \texttt{ARG1}, \texttt{EQ}\rangle$, and $\langle 3, 1, \texttt{ARG1}, \texttt{NEQ}\rangle$, the function would return two sets: $(\{0, 1, 2\}, \{3\})$.

[9] It is stateful in that each call increments an internal counter; if calling NewVar(`h`) returns `h1`, then the next call NewVar(`x`) returns `x2`, etc.

Algorithm 4 Converting from DMRS to MRS

Input: N is a list of Nodes as $\langle id, pred, Prop, carg \rangle$ tuples
Input: L is a list of Links as $\langle from, to, role, post \rangle$ tuples
Output: R is a list of Elementary Predications as $\langle pred, label, var, Arg \rangle$ tuples
Output: HC is a list of Handle Constraints as $\langle lhs, rel, rhs \rangle$ tuples

function DmrsToMrs(N, L)
    ▷ Step 1: Pre-assign labels to node ids
    for all $conj \in$ ConnectedEqComponents(L) **do**
        $h \leftarrow$ NewVar("h")
        for all $id \in conj$ **do**
            $H_{id} \leftarrow h$
    ▷ Step 2: Initialize EPs
    for $i \leftarrow 1, |N|$ **do**
        $R_i^{pred} \leftarrow N_i^{pred}$
        $R_i^{label} \leftarrow H_{id}$, where $id = N_i^{id}$
        if N_i is not quantifier **then** ▷ Set bound variable of quantifier later
            $R_i^{var} \leftarrow$ NewVar(N_i^{Prop}("cvarsort"), $\{\langle k, v \rangle \in N^{Prop} : k \neq$ "cvarsort"$\}$)
        if N_i^{carg} is not nil **then**
            R_i^{Arg}("carg") $\leftarrow N_i^{carg}$
    ▷ Step 3: Assign EP arguments and make qeqs
    for all $\{l \in L : l^{role} \neq$ "MOD"$\}$ **do**
        $i \leftarrow$ index of node where $N_i^{id} = l^{from}$
        $j \leftarrow$ index of node where $N_j^{id} = l^{to}$
        if $l^{post} \in \{$"NEQ", "EQ"$\}$ **then**
            $R_i^{Arg}(l^{role}) \leftarrow R_j^{var}$
        else if $l^{post} =$ "HEQ" **then**
            $R_i^{Arg}(l^{role}) \leftarrow R_j^{label}$
        else if $l^{post} =$ "H" **then**
            $hole \leftarrow$ NewVar("h")
            $R_i^{Arg}(l^{role}) \leftarrow hole$
            append $\langle hole,$ "qeq", $R_j^{label} \rangle$ to HC
            if $l^{role} =$ "RSTR" **then** ▷ Now set bound variables of quantifiers
                $R_i^{var} \leftarrow R_j^{var}$

The last step, step 3, now assigns the role arguments on the EPs based on information from the links. Label equalities have already been set in steps 1 and 2, so there is no need to consider MOD/EQ links. Furthermore, all other links with *post* values of `EQ` are now treated the same as those with `NEQ` as the label equality information is already accounted for. There are therefore three ways of interpreting a link, based on the value of *post*. Those with the values `EQ` or `NEQ` are regular arguments, so the EP's argument value will be the intrinsic variable of the EP corresponding to the *to* node. Those with a `HEQ` value select the label of the EP corresponding to the *to* node. And those with a `H` value create a new *hole* variable and a qeq such that the hole is the *lhs* of the qeq, and the *rhs* is the label of the EP corresponding to the *to* node. In that third case, `RSTR/H` links are used for quantifiers, so I take this opportunity to set the bound variable of the quantifier to the intrinsic variable of the EP corresponding to the *to* node.

This process converts DMRSs to MRSs, but it is simplified for clarity here. In the actual procedure implemented by PyDelphin, there are other node properties copied to EPs, such as for surface alignments, as well as the creation of a TOP handle and its corresponding qeq to the label of the top EP.

5.3 Semantic Tests

There are several properties of semantic structures that can influence how I can use or work with the representation. In Section 5.3.1 I explain the notion of **connectedness** in MRS and DMRS, since there is little I can do with a disconnected structure. When comparing two subgraphs, such as the source and target representations in bilingual data, the property of **isomorphism**, described in Section 5.3.2, is a useful indicator that the representations contain the same internal structure. And when traversing semantic graphs, **link orientation**, described in Section 5.3.3, is a property that helps me determine the direction in which to traverse edges.

5.3.1 Connectedness

MRS, DMRS, and many other semantic representations are graphs, and therefore can be disconnected, where some subgraph of semantic information is not incorporated into the greater structure. Fully connected representations are generally preferred, as they can be inspected via graph traversals to understand the contribution of each subgraph to the overall meaning of the representation. Also, some methods of serialization cannot accommodate disconnected graphs, such as PENMAN (see Section 5.5). In this section, I describe methods for determining if MRS and DMRS representations are connected or not.

An MRS can be considered connected if the EPs, taken as nodes, are connected. EP connectedness is determined by the following criteria:

1. if EP R_i has a non-scopal argument selecting EP R_j, then R_i and R_j are connected

2. if EPs $R_j \ldots R_k$ share a label, all EPs $R_j \ldots R_k$ are connected to one another

3. if EP R_i has a scopal argument that selects the label of EPs $R_j \ldots R_k$ (either directly or via qeq), then R_i is connected to each of $R_j \ldots R_k$

It is not unreasonable to add an additional criterion that the EP graph is connected to TOP, as some implementations (e.g., EDS) or algorithms may rely on this property, but in this dissertation, as in PyDelphin, I do not require it for MRS connectedness. Criterion 2 is somewhat controversial, as some may not consider collocation in the scope tree to be as strong a relationship as one being the argument of the other.

In DMRS, scope information has been made into explicit links where required, and the resulting nodes and links make up simpler graphs than those made up of MRS EPs, arguments, handle constraints, and variables. It is thus simpler to check for DMRS connectedness: just see if there's a spanning graph by traversing the links. Checking DMRS connectedness by traversing links is equivalent to checking MRS connectedness by traversing arguments and EP conjunctions, and a connected MRS is convertible to a connected DMRS.

5.3.2 Isomorphism, Structural Isomorphism, and Bilingual Isomorphism

Sometimes two equivalent MRSs can differ in their linearized form, which makes the comparison of two MRSs by their serialized strings inaccurate. For instance, a variable (e.g., `x5`) used in one MRS may have a different form from one (e.g., `x8`) used in another MRS, but as long as the variable type and distribution of these variables is the same in both MRSs, the meaning would be equivalent, assuming there are no other more significant differences. For DMRS, which has no variables, the value of node identifiers is similarly useful for its distributional qualities, but is otherwise meaningless. These differences can occur from using a different processor, such as the LKB instead of ACE, or from parsing close paraphrases where the word order is different but the meaning is the same. When form-only differences such as these occur, an **isomorphism** test can determine if the meaning of two representations is the same. Isomorphism checks if the shape of two graphs are the same, where meaningful content (defined below) is used to find corresponding nodes and edges.

For both MRS and DMRS, the values of predicates and constant argument values (as node labels) and argument roles (as edge labels) are meaningful content. MRS additionally includes handle constraint relations (namely, *qeq*), and DMRS includes post-slash labels on links. Variable properties are also generally included, although for some applications it can be useful to ignore them so sentences such as *the dogs chase cats* and *the dog chased cats*, which only differ by number and tense properties, are evaluated as isomorphic. Variables or node identifiers, surface alignments, or string differences in serialization (spacing, relative order of EPs/nodes, etc) are irrelevant for meaning, and thus also when computing isomorphism.

PyDelphin computes isomorphism via the VF2 algorithm (Cordella et al., 2001) by converting its internal MRS and DMRS representation into a NetworkX (Hagberg et al., 2008) graph and using NetworkX's isomorphism implementation.[10] The graph conversion includes only the relevant information as defined in the preceding paragraph.

For the work described by this dissertation, I use a relaxed form of isomorphism testing

[10] At least, for PyDelphin versions `0.3` through `0.6.2`.

for bilingual representations, which I call **structural isomorphism**.[11] For this test, I ignore predicate forms, as I am interested in checking for structural isomorphism only, and I ignore variable properties. Since predicate forms and variable properties do not matter, I implement this as a string comparison, which is faster to compute than the graph-based comparison, but requires decisions in serialization to be deterministic. I start with the PENMAN-serialized DMRS (see below in Section 5.5), which has a deterministic form for serialization, including normalized node identifiers, and remove substrings related to predicates, constant arguments, properties, and spacing, leaving only the bracketing, node identifiers, and argument roles. For example, Fig. 5.4 shows a simple transformation of the PENMAN serialization of the DMRS graph for *dogs chase cats*.

- Before: `(e0 / _chase_v_1 :ARG1 (x1 / _dog_n_1) :ARG2 (x2 / _cat_n_1))`
- After: `(e0:ARG1(x1):ARG2(x2))`

Figure 5.4: Before and after the string transformation for structural isomorphism checking

For two subgraphs in a bilingual pairing that have the same structural form as determined by this string transformation, I say that they are **bilingually isomorphic**, and this property is evidence that the predicates at corresponding locations in the subgraphs are likely to be translationally equivalent. In a strict sense, there are some problems with the conclusions I draw from this test's outcome. First, the two subgraphs are not guaranteed to be translationally equivalent in whole or in part, but I make that assumption by trusting the output of my bilingual subgraph aligners (see Chapter 6). Second, even if the subgraphs are translationally equivalent, it is not necessarily the case that corresponding nodes in the graph are also translationally equivalent. There could be argument switching (of the *I likes apples* versus *apples please me* type), head switching (*I like swimming* versus *I swim hap-*

[11] Thanks to Emily Bender for the term *isomorphish* to describe a relationship as somewhat isomorphic, but I will use *structural isomorphism* here as it emphasizes the comparison of graph structure.

pily), or other kinds of translational divergences (Dorr (1994) gives a thorough overview of such divergences, which I covered in Section 1.1) that still fit within my notion of bilingual isomorphism. Nevertheless, I find the test to be generally useful in approximating structural correspondence, so I use it when converting subgraph pairs to MRS transfer rules in Chapter 7.

5.3.3 Link Orientation and Inversion

DMRS is a representation of semantic dependencies and in dependency relations there are **heads** and **dependents**.[12] These correspond to functors and arguments, which in DMRS links are encoded as *from* (for the head, selecting the functor) and *to* (for the dependent, selecting the functor's argument). The *from*→*to* sequence encodes a link's **direction** in the semantic graph. Using the link direction for a directed graph traversal would, however, lead to multiply rooted structures and hence unreachable portions for nearly all DMRS graphs. Consider the DMRS in Fig. 5.5 for the sentence *the southbound train departed*, which has three roots: `_the_q`, `_southbound_a_1`, and `_depart_v_1`. A directed traversal starting at the indicated top node (`_depart_v_1`) would only be able to reach `_train_n_of`.

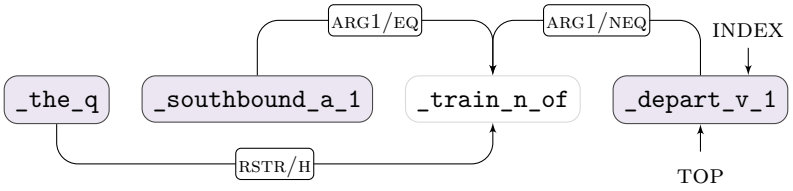


Figure 5.5: Multiply-rooted DMRS for *The southbound train departed.*

By **inverting** the direction of traversal of certain links, the multiply-rooted graphs can become singly-rooted. Discussion of traversal and the useful properties of singly-rooted graphs is in Section 5.4.2; here I define the test I use for determining when to invert the

[12]These are properties of edges, not nodes, in the dependency graph, as a node can be a head for one relation and a dependent of another.

direction of link traversal. The functor-argument information in link direction should be maintained, as it is important for the well-formedness of the semantic structures, so I distinguish link direction from the preferred direction of edge traversal with a new term: **link orientation**. Links that are preferably traversed *from*→*to* (i.e., the same as the link direction) are ***to*-oriented** while those preferably traversed *to*→*from* are ***from*-oriented**. By the *preferred* direction of traversal, I mean I designate links as *to*-oriented or *from*-oriented by their attributes, not their usage—in other words, the orientation need not always be the actual direction of traversal. Specifically, links from quantifiers to their quantifiees and from non-scopal modifiers to their modifiees are generally those causing multiple roots and thus designated *from*-oriented. The quantifier links (e.g., `_the_q` to `_train_n_of`) have a *role* of RSTR which is used in no other kind of link. These links invariably have a *post* of H, so I will refer to them as RSTR/H links. Non-scopal modifiers (e.g., `_southbound_a_1`) may be the *from* node of more than one link (i.e., in MRS, they have more than one argument), but only one is connected to the modifiee: the one with the *post* of EQ (in MRS, they share a label). These links occur with different *role* values, so I call them */EQ links.

Algorithm 5 defines the test for link orientation—*to*-orientation, specifically—by returning False if either the link's *role* is RSTR or its *post* is EQ, otherwise returning True. Link orientation is dichotomous, so a False value for *to*-orientation is equivalent to a True value for *from*-orientation. I therefore only need one test to determine both orientations.

Algorithm 5 Determine if a DMRS link is *to*-oriented

Input: l is a Link as a $\langle$*from*, *to*, *role*, *post*$\rangle$ tuple
Output: True if l is *to*-oriented, otherwise False

```
1 function ToOrientedLink(l)
2     if l^role = "RSTR" or l^post = "EQ" then
3         return False
4     return True
```

The quantifier case is straightforward but non-scopal modifiers are more nuanced and require further explanation. Note that for the DMRS figures below I highlight the *from*-

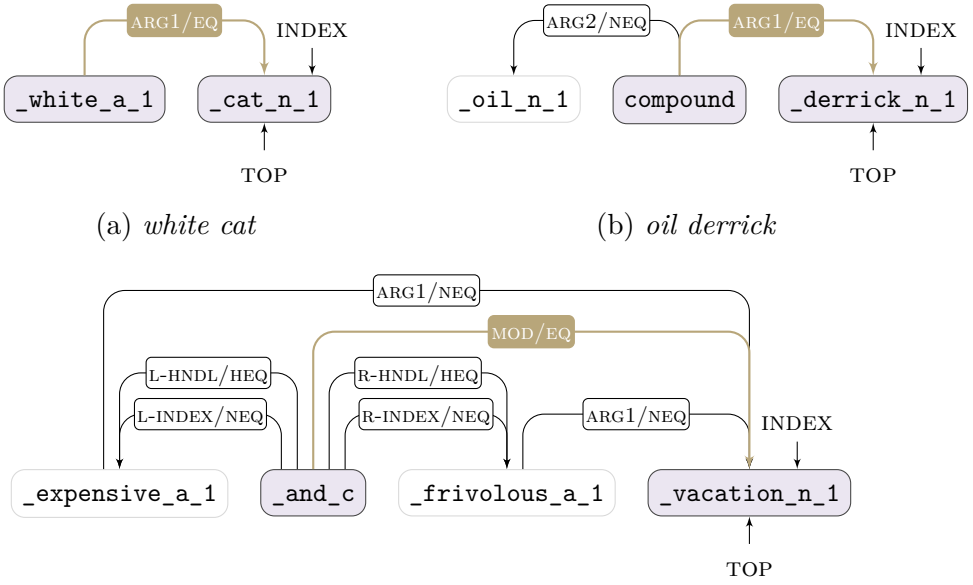
oriented links that would be inverted in a traversal. Non-scopal modifiers are generally things like intersective modifiers (e.g., *white* in *white cat*, as shown in Fig. 5.6a), but also subsective (*cognitive scientist*), intensional (*fake gun*), and so on, where the modifier is connected to the modifiee with an ARG1/EQ link. Some modifiers are composed of multiple nodes, such as noun-noun compounds (e.g., *oil derrick*, shown in Fig. 5.6b), which introduce a node with the abstract predicate `compound` and an ARG1/EQ link that connects `compound` to the node being modified by the compounding (*derrick*). Another link connects `compound` to the node for the modifying noun (*oil*), but its *post* is not EQ. Thus, the subgraph composed of `compound` and the modifying noun together act as a modifier of the other noun in the compound. Another example is coordinated adjectives (e.g., *expensive and frivolous vacation*, shown in Fig. 5.6c),[13] where the coordinator node (with the `_and_c` predicate), rather than the adjectival coordinands, has a */EQ link connecting it to the modifiee, specifically a MOD/EQ link.

The */EQ links are not used for **scopal modification**, such as *seemingly difficult* (e.g., in *the seemingly difficult problem was solved quickly*; the relevant portion, *seemingly difficult problem*, is shown in Fig. 5.7a, compared to the non-scopal modification of *difficult problem* in Fig. 5.7b). Scopal modifiers link to their scopal-modifiees with a *post* value of H (in MRS, they have an argument that is qeq to a label scoping over the modifiee). Note that the subgraph composed of the scopal modifier and its modifiee together non-scopally modify the noun (e.g., *problem*, in the example above) via a MOD/EQ link. The linguistic intuition here is that the *problem* in *seemingly difficult problem* is not *difficult* (hence there is no non-scopal modification between the nodes for *difficult* and *problem*), it is only *seemingly difficult*.

Link orientation is related to the notion of representative nodes in MRS to DMRS conversion (see Section 5.1.2). In MRS, a representative EP will be take no other EPs in its EP conjunction as arguments. In DMRS, all */EQ links involving a representative node will have the representative node as its *to* node, including the non-argument MOD/EQ links.

---

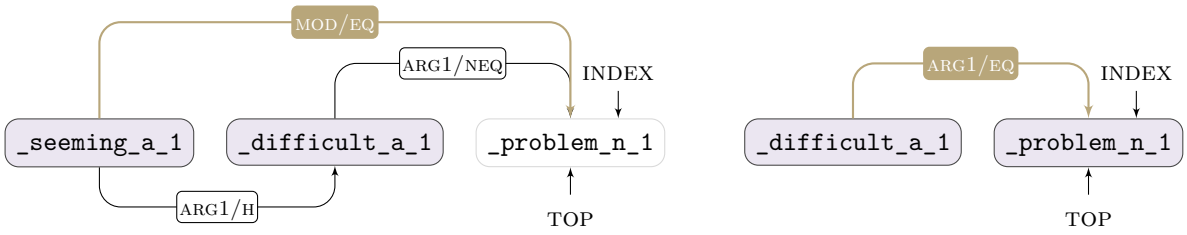
[13] Also see the DMRS for *the large and gentle dog sleeps* in Fig. 2.5e of Section 2.10.



(a) *white cat* (b) *oil derrick*

(c) *expensive and frivolous vacation*

Figure 5.6: DMRS subgraphs showing non-scopal modification



(a) *seemingly difficult problem* (b) *difficult problem*

Figure 5.7: DMRS subgraphs comparing (a) scopal and (b) non-scopal modification

Thus, in a rooted traversal, the representative node will be the root and the *from* node of all */EQ links will be its descendants.

5.3.4 Summary

The three tests above—connectedness, isomorphism, and *to*-orientation—are useful for filtering, comparing, and traversing semantic graphs. I use them for tasks including extracting subgraphs in Section 5.6, bilingual node alignment in Section 7.3.2, rooted graph traversals in Section 5.4.2, and PENMAN serialization in Section 5.5.

5.4 Semantic Graph Traversal

Here I describe methods of traversing semantic graphs. The first method (Section 5.4.1) is an iteration over only the nodes of a DMRS, which is used later to prepare my data for use in an n-gram aligner (see Section 6.1). The second method (Section 5.4.2) is a *to*-oriented traversal of the full DMRS graph, which is used in PENMAN serialization (see Section 5.5).

5.4.1 Surface Order Node Iteration

Surface-order node iteration enumerates the nodes based on the order of their corresponding tokens in the sentence. This link-independent method may traverse from node N_i to N_j even if there are no links $\langle N_i, N_j, *, * \rangle$ or $\langle N_j, N_i, *, * \rangle$, and it will visit every node in the graph. Processors such as ACE reliably output nodes in surface order by default, so for my experiments I use the order as output from the processor. If, however, a semantic representation is obtained such that the node order is not reliable, the following method can be used to recover this order in most circumstances. Some grammars may contain bugs such that some pieces of information required for this method (such as surface alignments) are not emitted with the semantic representation, in which case the full recovery of the surface order is impossible. For this reason, it is preferable to use the predicate ordering from a processor like ACE, rather than always recomputing the order.

I define this method as a sorting operation on nodes. Recall from Section 2.2.3 that the *MRS representations often store a ⟨CFROM, CTO⟩ (i.e., start and end) pair of pointers to surface positions along with EPs/nodes and this is the primary information that can be exploited for sorting the nodes. If all nodes corresponded to non-overlapping tokens, this would simply be the ordering of the starting position (CFROM). However, there are some situations where a predicate's surface pointers overlap with one or both pointers from other predicates, so I must make use of other kinds of information to break ties.

The first sorting criterion is the starting position of each node's surface pointers (CFROM). But consider a noun compound like *machine translation*. The nodes and surface pointers, as analyzed by the ERG, are shown in Table 5.1.[14] Note that implicit quantifiers (`udef_q`) overlap exactly with the thing they quantify over, and the abstract `compound` node overlaps with all other nodes in the compound.[15]

node	CFROM	CTO
`udef_q`	0	19
`compound`	0	19
`udef_q`	0	7
`_machine_n_1`	0	7
`_translation_n_1`	8	19

Table 5.1: Nodes and surface pointers for *machine translation*

An improvement over just ordering by the starting position is to order first by CFROM, then break ties by the inverse or negation of CTO, which would put the outermost (i.e., widest-ranging) nodes first. But this still does not result in a deterministic ordering of the implicit quantifiers, so therefore I add a third-level tie-breaker: node-is-a-quantifier (or

[14]The relative ordering of nodes in Table 5.1 and successive tables is as output from the ACE processor.

[15]The first `udef_q` quantifies over the compound so it takes the full span.

not),[16] which recovers the order in Table 5.1.

Consider now the ERG's analysis of the negation suffix *n't*, as in *Kim isn't a student*, which results in an abstract node `neg`, and not a surface node (e.g., some hypothetical `_n't_a_neg`). In contrast, the analysis for the prefix *un-*, as in *Kim unwound the string*, results in a surface node `_un-_a_rvrs`, rather than an abstract one (e.g., `rvrs`). The nodes and pointers for *isn't* and *unwind* are shown in Table 5.2.

node	CFROM	CTO
`_be_v_id`	4	9
`neg`	4	9
`_wind_v_1`	4	11
`_un-_a_rvrs`	4	11

Table 5.2: Nodes and surface pointers for *isn't* in *Kim isn't a student* and *unwound* in *Kim unwound the string*

While the *n't* suffix and *un-* prefix have a relative order in the sentence, their surface pointers in the semantic representation are shared with the words they modify. A fourth-level criterion such as node-is-abstract would not place these in the desired position (neither node is abstract for *un-*), and other solutions (e.g., alphabetic sort) seem arbitrary or brittle, so it does not seem possible to have a principled criterion for fully surface-sorted nodes given only the information in the semantic graph. Therefore I opt for consistency over principle and add both criteria above (node-is-abstract and alphabetic sort) as fourth- and fifth-level tie-breakers.

[16]This criterion may need to be adapted for different languages. In English and Japanese, overt quantifiers precede the things they quantify over (e.g., *the dog*; or この犬 *kono inu* "this dog", where the demonstrative *kono* is the quantifier), so it is consistent to put the implicit quantifiers in the same relative position, but in, e.g., Indonesian, quantifiers follow the quantifiees (e.g., *anjing ini* "this dog", where the demonstrative *ini* is the quantifier), so the consistent placement of implicit quantifiers is after the quantifiee.

5.4.2 Rooted Traversal

In Section 5.3.3 I defined a test to determine if a link is *to*-oriented (i.e., the preferred traversal of the link matches its *from*→*to*, or functor-argument, direction) or *from*-oriented (the preferred traversal is *to*→*from*), which lets me invert the edges of quantifier and non-scopal modifier links so that the multiply-rooted DMRS graphs become singly-rooted. In this section I will describe that traversal. A similar exploration on MRS graphs has been called **argument crawling** (Packard et al., 2014), but it did not use link-orientation. Lønning and Oepen (2016) call a similar procedure for recovering scopal information in EDS **spinal traversal**.

The traversal is defined recursively starting from the TOP node. Upon visiting a node N_i, the procedure first collects all *to*-oriented links where N_i is the *from* node and calls itself on their *to* nodes. Next it collects all *from*-oriented links representing non-scopal modification where N_i is the *to* node and calls itself on their *from* nodes. Finally it finds all *from*-oriented representing quantification and calls itself on their *from* nodes. The iteration order of the *to*-oriented links is the order the roles appear in the DMRS, e.g., with ARG1 appearing before ARG2, etc. For the non-scopal modifiers, it is possible that many edges share the same label (e.g., ARG1/EQ′), so this subset is ordered instead by their surface alignments.[17]

Fig. 5.8 shows the DMRS for the sentence *The angry farmer fanatically chased the rabbit.*[18] Note that there are more roots (`_the_q`, `_angry_a_1`, `_fanatic_a_1`, and `_the_q`) than non-roots (`_chase_v_1`, `_farmer_n_1`, and `_rabbit_n_1`), and the indicated top of this graph (`_chase_v_1`) is not even a root node. A traversal of the graph that inverts the *from*-oriented edges results in the graph shown in Fig. 5.9. Inverted edges are marked with a prime symbol (′), and numbers in parentheses indicate the order of traversal using a depth-first search.

It is not guaranteed that a traversal using such link inversions would reach every node. Links that prevent graphs from being fully traversable occur when *to*-oriented links are

[17] As mentioned in Section 5.4.1, I use the surface order as output by ACE.

[18] This visualization is a variation of the graphical views of DMRS that emphasizes the edge directions.

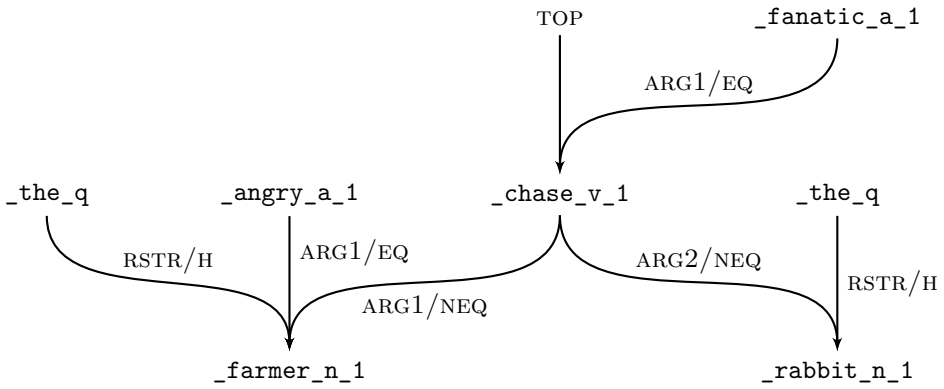


Figure 5.8: Multiply-rooted DMRS for *The angry farmer fanatically chased the rabbit*

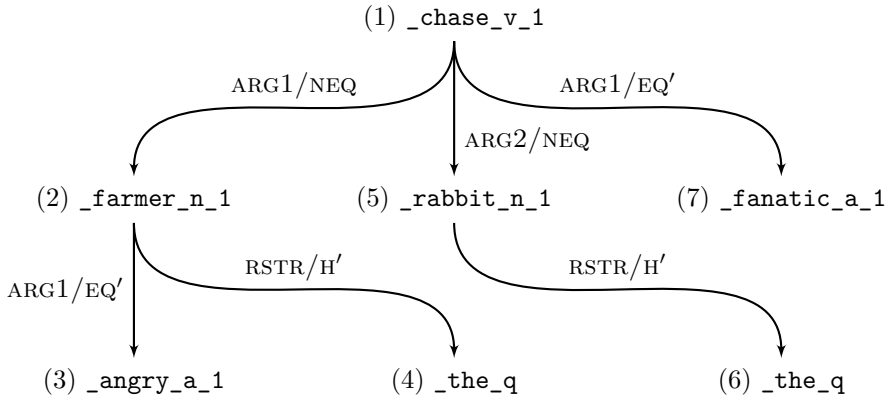


Figure 5.9: Rooted DMRS traversal order with inverted links

directed toward the top, or root, rather than away from it, and likewise when *from*-oriented links point away from the root, rather than toward it. For the development portion of the Tanaka Corpus (Tanaka 2001; also see Section 8.1.1), 90.4% of the Japanese items analyzed with Jacy had a result that could be fully traversed using only the preferred traversal directions (79.3% over all results). For the English results, as analyzed by the ERG, it was 99.4% of the items (98.1% over all results). Preliminary analysis of the results that cannot be fully traversed shows them to be ill-formed in some way, indicative of a bug in the grammar, but a full analysis is beyond the scope of this dissertation. Such a use of semantic inspection is

not without precedent, however. Fuchss et al. (2004) described MRS representations in terms of **dominance constraints** and defined **MRS-nets** as MRSs that satisfy some graphical properties similar to my requirements for a full traversal using preferred traversal directions, and Flickinger et al. (2005a) showed that analyses that were not MRS-nets indicated bugs in the grammar.

A DMRS graph or subgraph that can be fully traversed has some useful properties. The traversal changes the multiply-rooted graph into a singly-rooted graph, but the graph is in fact often an **arborescence**, or a directed, rooted tree. Algorithms designed for trees are generally more efficient than those for less constrained types of graphs, which is appealing for computational tasks involving millions of subgraphs, such as my transfer rule extraction. Furthermore, subgraphs of trees (i.e., subtrees) can be found by pruning just one edge. Singly-rooted DMRS graphs are not always arborescences, though, as re-entrancy is a common feature in *MRS representations, e.g., in control examples like *Kim tried to sleep* (where *Kim* is both trying and sleeping), but is also seen in the examples above, namely Figs. 5.2, 5.6c, and 5.7a. Subgraphs with rentrancies cannot be found by pruning a single edge when the edge occurs between, in an undirected sense, the source and target of the re-entrancy, but it is possible if the edge is outside the undirected cycle caused by the re-entrancy.[19] The rooted traversal also has a normalizing effect on source and target semantic representations. Nodes at similar locations in the source and target graph typically have similar significance or salience with respect to the overall meanings; i.e., nodes that are more distance from the root are less important for representing the meaning of the sentence. I make use of this fact by assuming that the root nodes of aligned subgraphs are translationally equivalent and flag them as such so the transfer process can copy any untransferred material (e.g., morphosemantic properties, which I do not cover) from the source root to the target root (see Section 7.3.3).

[19]This fact could be exploited as a feature or a constraint of subgraph enumeration, but I do not explore that possibility in this dissertation.

5.4.3 Summary

I have defined two methods for traversing DMRS structures. The first, defined in Section 5.4.1, iterates over nodes in the DMRS according to their surface order. I describe a way to compute that ordering from graph properties in case it is necessary, but in practice I use the output order from ACE. The second, defined in Section 5.4.2, uses the links' orientations to choose a preferred direction of traversal. This method is not guaranteed to reach all nodes, but it does for nearly all well-formed DMRSs. These traversal methods are useful for preparing data for bilingual alignment in Chapter 6.

5.5 PENMAN Serialization

Converting MRS to DMRS simplifies the semantic structure to aid in discovering useful information. The DMRS graph, however, is still challenging to work with in some ways, such as its being a multi-rooted DAG. Performing a task like graph comparison as a graph operation can be prohibitively slow, but if equivalent graphs could be consistently serialized to the same string, they could be compared with a much-faster string comparison operation. Therefore I define a simple serialization of the graphs along with a traversal method that visits the nodes in a well-defined order. For the serialization format, I chose to use the PENMAN notation,[20] which has in recent years been popularized by the AMR data (Banarescu et al., 2013). PENMAN notation is designed for encoding something like semantic dependencies, so it is a good fit with the requirements of DMRS. Furthermore, by adopting the existing format, code written for AMR data can use DMRS data with little or no modification, which lowers barriers for researchers outside of DELPH-IN wanting to make use of DELPH-IN data.

The first step of the conversion process is transforming DMRS nodes and links into ⟨*source*, *relation*, *target*⟩ triples, as shown in Algorithm 6. The *source* and *target* of the triples are edge directions, not the *from* and *to* of DMRS links, but the functor-argument information is not lost because the *relation* indicates if the direction has been inverted,

[20]Historically called **Sentence Plan Notation** (Kasper, 1989) in the PENMAN project (Penman, 1989)

as described below. The top of the graph is important, as PENMAN is a singly-rooted serialization, so the top of the DMRS is identified with a special triple. Next, all attributes of nodes are added as individual triples with the node's identifier as the source. Links are added last; all *to*-oriented links are added as-is (as ⟨*from*, *relation*, *to*⟩ triples) while *from*-oriented links are inverted so as to be in the preferred traversal direction (as ⟨*to*, *relation*-***of***, *source*⟩ triples).[21] An example of the output of this algorithm for *the dog whose tail is long barked* is given in Fig. 5.10, with the node-derived triples on the left and the link-derived triples on the right.

Algorithm 6 Converting DMRS nodes and links to triples

Input: *top* is the identifier of the top node in the DMRS
Input: N is a list of Nodes as ⟨*id*, *pred*, *Prop*, *carg*⟩ tuples
Input: L is a list of Links as ⟨*from*, *to*, *role*, *post*⟩ tuples
Output: T is a list of ⟨*source*, *relation*, *target*⟩ triples

```
1  function DmrsTriples(top, N, L)
2      append ⟨"top", "TOP", top⟩ to T                ▷ Identify the top node of the graph
3      for all n ∈ N do
4          append ⟨n^id, "predicate", n^pred⟩ to T
5          if n^carg is not nil then
6              append ⟨n^id, "carg", n^carg⟩ to T
7          for all ⟨k, v⟩ ∈ n^Prop do
8              append ⟨n^id, k, v⟩ to T
9      for all l ∈ L do
10         if ToOrientedLink(l) then
11             relation ← FormatString("%s-%s", l^role, l^post)        ▷ E.g., ARG1-NEQ
12             append ⟨l^from, relation, l^to⟩ to T
13         else
14             relation ← FormatString("%s-%s-of", l^role, l^post)     ▷ E.g., ARG1-EQ-of
15             append ⟨l^to, relation, l^from⟩ to T
```

Once I have the triples, I can serialize to PENMAN notation by starting at the identified top node adding all triples with that node as the source, then recursively expanding triple

[21] The relations of link-derived triples combine the *role* and *post* values of the links. I use FormatString() in Algorithm 6 to represent a string-formatting function with the semantics of C-style `printf` functions.

```
(top, TOP, 10006)                    (10001, RSTR-H-of, 10000)
(10000, predicate, _the_q)           (10001, ARG2-EQ-of, 10003)
(10001, predicate, _dog_n_1)         (10001, MOD-EQ-of, 10005)
(10001, cvarsort, x)                 (10003, ARG1-NEQ, 10004)
(10002, predicate, def_explicit_q)   (10004, RSTR-H-of, 10002)
(10003, predicate, poss)             (10005, ARG1-NEQ, 10004)
(10003, cvarsort, e)                 (10006, ARG1-NEQ, 10001)
(10004, predicate, _tail_n_1)
(10004, cvarsort, x)
(10005, predicate, _long_a_1)
(10005, cvarsort, e)
(10006, predicate, _bark_v_1)
(10006, cvarsort, e)
```

Figure 5.10: DMRS triples for *the dog whose tail is long barked*

targets as new nodes. The details of the layout algorithm are mostly irrelevant here (I rely on my Penman library[22] for serialization), but it is important to note that it is deterministic. It is not strictly necessary to invert the *from*-oriented links in Algorithm 6, but by doing so I am informing the serialization function in the Penman library of the preferred direction for those edges. The library will respect these preferences unless it is unable to reach portions of the graph, in which case it opportunistically switches one edge direction and proceeds as before.[23] For each node, triples with the `predicate` relation (abbreviated as `/`) are always traversed first, followed by those with `carg` and property relations, then uninverted triples from the node (*to*-oriented), and finally the inverted triples from the node (*from*-oriented). It is possible to sort the relative order of property- and link-derived triples, but I keep the original order from the DMRS. The result of serializing the DMRS given above in Fig. 5.2 is shown in Fig. 5.11.[24]

For my work on bilingual semantic alignment, the default serialization in Fig. 5.11 is

[22] `https://github.com/goodmami/penman`

[23] Most DMRS graphs can be serialized fully with the original edge directions, but some instances require inverted edges in order to get a spanning DAG.

[24] As in Fig. 5.2, I omit most node properties.

```
(10006 / _bark_v_1
    :cvarsort e
    :ARG1-NEQ (10001 / _dog_n_1
        :cvarsort x
        :RSTR-H-of (10000 / _the_q)
        :ARG2-EQ-of (10003 / poss
            :cvarsort e
            :ARG1-NEQ (10004 / _tail_n_1
                :cvarsort x
                :RSTR-H-of (10002 / def_explicit_q)))
        :MOD-EQ-of (10005 / _long_a_1
            :cvarsort e
            :ARG1-NEQ 10004)))
```

Figure 5.11: PENMAN serialization for *the dog whose tail is long barked*

made even simpler by combining the `cvarsort` property with the node identifier.[25] I also renumber the identifiers via a depth-first traversal, which gives a deterministic identifier for each node. The result is shown in Fig. 5.12.

```
(e0 / _bark_v_1
    :ARG1-NEQ (x1 / _dog_n_1
        :RSTR-H-of (u2 / _the_q)
        :ARG2-EQ-of (e3 / poss
            :ARG1-NEQ (x4 / _tail_n_1
                :RSTR-H-of (u5 / def_explicit_q)))
        :MOD-EQ-of (e6 / _long_a_1
            :ARG1-NEQ x4)))
```

Figure 5.12: ID-normalized PENMAN serialization for *the dog whose tail is long barked*

PENMAN notation is a compact way of serializing DMRS graphs. The compactness saves disk space when storing millions of subgraphs (e.g., during bilingual subgraph alignment in Chapter 6) and the deterministic node relabeling and serialization allows subgraphs to be compared via string comparisons. Graph comparison is generally expensive to compute, so

[25] Quantifiers, which have no `cvarsort`, are assigned the (underspecified) `cvarsort` value of `u`.

the ability to do string comparisons instead allows my subgraph alignment methods to be tractable.

5.6 Subgraph Extraction

For bilingual semantic alignment, the ability to extract subgraphs from the original semantic graph is necessary functionality. I extract subgraphs in two ways: from a selection of nodes or from a restricted graph traversal. The first method is used when the subgraph's predicates are known, while the second is used to explore the space of available subgraphs for a semantic representation.

5.6.1 Subgraphs From a List of Predicates

A DMRS graph contains uniquely identifiable nodes (via their node identifiers) which are characterized primarily by their predicates. A single DMRS instance may have multiple nodes with the same predicate, so the use of predicate names to identify nodes is an inexact specification. The method of subgraph extraction described in this section finds all subgraphs connecting nodes selected by a **predicate phrase**, i.e., a list of predicates. For every unique predicate occurring n times in the list of predicates P, the resulting subgraphs will each have n nodes with that predicate. There may be more than one subgraph found in a DMRS for each predicate list, but only connected subgraphs representing unique node selections will be returned. The method is defined in Algorithm 7.

The first step is to find, for each predicate P_i, the set of nodes M_i using that predicate (line 3). If there are repeated predicates in P then there will be repeated sets in M, but this is by design, as I want one unique node for every predicate. The next step (line 4) takes the Cartesian product of the list of node sets, finding all selections of nodes. For instance, if M is $(\{a, b\}, \{c\}, \{a, b\})$, there will be four selections: (a, c, a), (a, c, b), (b, c, a), (b, c, b). There is, however, only one of those selections that is useful. While a predicate may be repeated in a subgraph (e.g., for *violence begets violence*, or *the dog chased the cat*), a node may not. Also, node order is irrelevant in subgraph selection. Therefore, of the selections above, only

one of (a, c, b) or (b, c, a) are useful. I filter out repeated elements and repeated selections in line 5.

For each unique and complete node selection, there is exactly one subgraph as I include every link from the original DMRS whose *from* and *to* nodes both exist within the selection. Each subgraph must be tested for connectedness, as it is possible the node selection chooses nodes that are not connected via the link selection. Consider again *the dog chased the cat*: if P is (*the*, *cat*), the algorithm will make two node selections, one including the first *the* (determiner of *dog*) and one including the second, but only the latter selection is a connected subgraph. Therefore in line 8 I test if the node and link selections represent a connected subgraph. If the subgraph is connected, I append it to the subgraph list S.

Algorithm 7 Extracting DMRS subgraphs from a predicate list

Input: N is a list of Nodes as $\langle id, pred, Prop, carg \rangle$ tuples
Input: L is a list of Links as $\langle from, to, role, post \rangle$ tuples
Input: P is a list of predicates
Output: S is a list of connected DMRS subgraphs as $\langle nodes, links \rangle$ tuples

```
1 function DmrsSubgraphsFromPredList(N, L, P)
2     for i←1, |P| do                          ▷ Find sets of matching nodes for each P_i
3         M_i ← {N_j : (1 ≤ j ≤ |N|) and (N_j^pred = P_i)}
4     A ← M_1 × ··· × M_|P|                    ▷ All node selections for each predicate P_1, ..., P_|P|
5     U ← {N' ∈ A : no element N'_i = N'_j where i ≠ j}        ▷ Only unique selections

6     for all N' ∈ U do                        ▷ Extract subgraphs for each valid N' and L'
7         L' ← (l ∈ L : (l^from ∈ N'^id) and (l^to ∈ N'^id))
8         if Connected(N', L') then
9             append ⟨N', L'⟩ to S
```

5.6.2 Enumerated Subgraphs From a Traversal

The second method, defined in Algorithm 8, performs a rooted traversal of the graph from each node and records the depth of traversal at each node it reaches, then enumerates the subgraphs starting from each node, including each descendant node up to some maximum

depth. Unlike the method defined in Section 5.6.1, this algorithm does not require a list of predicates; it only requires the input semantic representation and a maximum depth.

The top function of Algorithm 8, DmrsEnumerateSubgraphs(), goes through each node in the input DMRS and, considering it the root of a class of subgraphs, finds all descendant nodes[26] and their depth, via rooted traversal, respective to the root. Subgraphs, from single nodes (depth 0) up to subgraphs of depth d, are enumerated and added to the list of subgraphs S. For re-entrant subgraphs, it is possible to enumerate the same subgraph more than once, at different depths, so in line 8 I check if the subgraph has been seen before adding it to S.

The second, recursive function DmrsNodeDepths() takes as parameters a current node n, the DMRS data, the current depth of traversal cd, and a set of already-traversed nodes *seen*. The base condition is that the current node has not yet been traversed, so it will continue until it cycles or until it runs out of links to traverse.[27] If the node has not yet been traversed, it and the current depth cd are added to the output node set N'. The algorithm then calls itself, at the next depth level, using the *to* node of any *to*-oriented links and the *from* node of any *from*-oriented links. All other links (where neither *from* nor *to* are the current node's *id*) are ignored.

The iteration over DMRS nodes as roots and the selection of nodes within a depth limit are something like the Root() and Frontier() operations used in Data-Oriented Parsing (e.g., Bod and Scha 2007) and Data-Oriented Translation (Way, 1999; Hearne and Way, 2003). In contrast, the method defined above works on graphs instead of trees. Furthermore, my method does not enumerate every possible subgraph, but one subgraph for each root and depth level (that is, it includes all nodes available within the depth limit, and not various subsets of them).

---

[26]For re-entrant graphs, a descendant could also be an ancestor, and this is by design. I avoid cycles in the traversal by stopping if I've encountered the same node twice.

[27]Dynamic programming and other optimizations, such as only traversing to the maximum depth, would improve the computational complexity, but I present it as it is here for simplicity.

Algorithm 8 Extracting DMRS subgraphs from a restricted graph traversal

Input: N is a list of Nodes as $\langle id, pred, Prop, carg \rangle$ tuples
Input: L is a list of Links as $\langle from, to, role, post \rangle$ tuples
Input: md is the maximum depth of traversal
Output: S is a list of connected DMRS subgraphs as $\langle Nodes, Links \rangle$ tuples

```
1  function DmrsEnumerateSubgraphs(N, L, md)
2      for all n ∈ N do
3          D ← DmrsNodeDepths(n, N, L, 0, {})
4          for d←0, md do
5              N' ← {n' : (⟨n', d'⟩ ∈ D) and (d' ≤ d)}
6              if |N'| > 0 then                          ▷ No empty subgraphs
7                  L' ← (l ∈ L : (l^from ∈ N'^id) and (l^to ∈ N'^id))
8                  if ⟨N', L'⟩ ∉ S then
9                      append ⟨N', L'⟩ to S
```

Input: n is the root Node as a $\langle id, pred, Prop, carg \rangle$ tuple
Input: N is a list of Nodes as $\langle id, pred, Prop, carg \rangle$ tuples
Input: L is a list of Links as $\langle from, to, role, post \rangle$ tuples
Input: cd is the current depth of traversal
Input: $Seen$ is a set of traversed Nodes
Output: N' is a set of (n', d) pairs where n' is a Node and d is the Node's depth from n

```
10 function DmrsNodeDepths(n, N, L, cd, seen)
11     N' ← {}
12     if n ∉ Seen then
13         add n to Seen
14         add (n, cd) to N'
15         for l ∈ L do
16             if l^from = n^id and ToOrientedLink(l) then
17                 tgt ← N_i, where N_i^id = l^to
18                 N' ← N' ∪ DmrsNodeDepths(tgt, N, L, cd + 1, Seen)
19             else if l^to = n^id and ¬ToOrientedLink(l) then
20                 tgt ← N_i, where N_i^id = l^from
21                 N' ← N' ∪ DmrsNodeDepths(tgt, N, L, cd + 1, seen)
```

5.6.3 Summary

The two subgraph extraction algorithms defined in Section 5.6.1 and Section 5.6.2 present different views of the data. One extracts subgraphs matching an external specification for included nodes, while the other enumerates subgraphs at various roots and sizes. These two methods are used in Chapter 6 to find bilingual subgraph alignments, the first method for when I have aligned semantic predicates, and the second for building a statistical model over the subgraphs.

5.7 DMRS Simplification

Even with the simpler DMRS graphs, there are still some kinds of information that are noise for bilingual aligners. This noise can be elements that are uninformative and increase data sparsity or, for n-gram aligners, tokens that intervene and prevent a useful n-gram from being found. Some of these substructures are entirely predictable, so it is possible to simply remove them from the graph. Some can be reduced to a form that simplifies the graph but doesn't lose information. And some represent information that is not necessarily useful, and can be dropped even if it cannot be recovered.

Fig. 5.13 shows the DMRS, as analyzed by the ERG and converted to DMRS by PyDelphin, for the phrase *Pierre Vinken, 61 years old*. It only has five words, but requires ten predicates, as each individual (the nodes for *Pierre*, *Vinken*, and *years*) requires a quantifier, the name requires a `compound` node to join them, and a `measure` node allows *61 years* to modify *old*. In this section, I will propose some transformations that reduce the graph to just five nodes, one for each word. Such transformations are not a new idea, as they have been done for DM (Ivanova et al., 2012; Oepen et al., 2014, 2015), and for DMRS (Yin et al., 2014; Copestake et al., 2016). I, however, am selective about which transformations to apply, sacrificing some complexity for a representation that is less lossy than those prior.

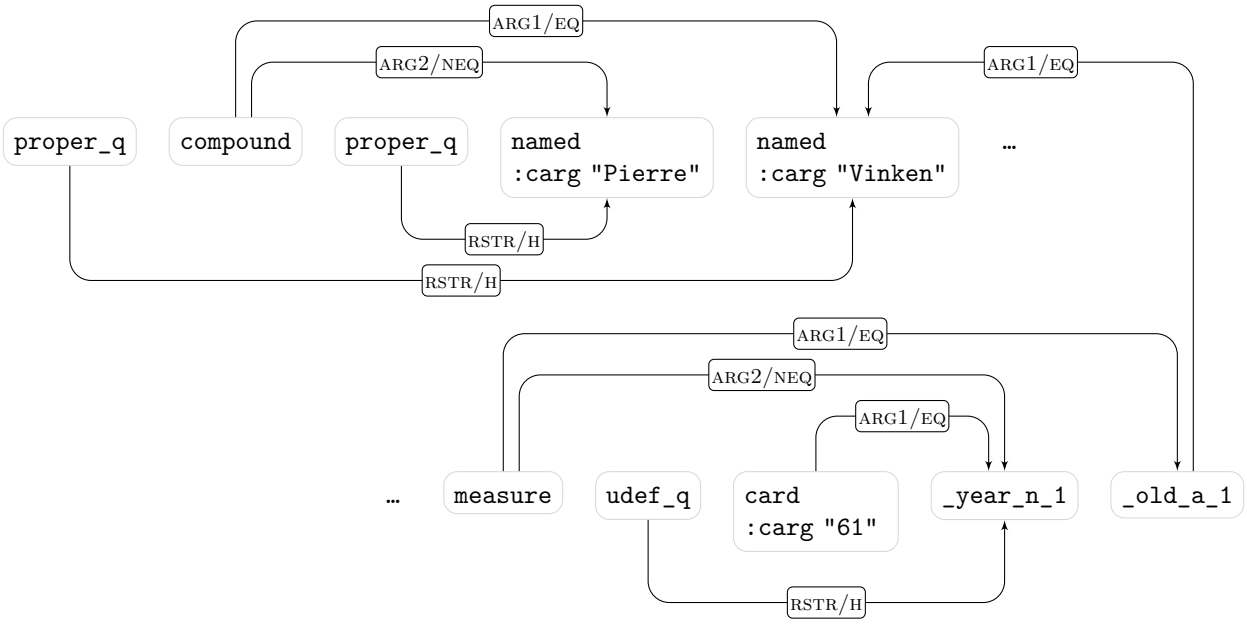


Figure 5.13: DMRS graph for *Pierre Vinken, 61 years old*

5.7.1 Removing Nodes

Some nodes in the DMRS graph can be removed and easily recovered. The nodes may be entirely predictable based on the presence of other nodes, or they may have been converted to links (see Section 5.7.3). Additionally, there are some kinds of irrecoverable nodes whose removal does not significantly change the meaning, such as the passivization marker `parg_d` in the ERG or the topic marker `_wa_d` in Jacy. In this section I will focus on the removal of one kind of generally recoverable node: default quantifiers.

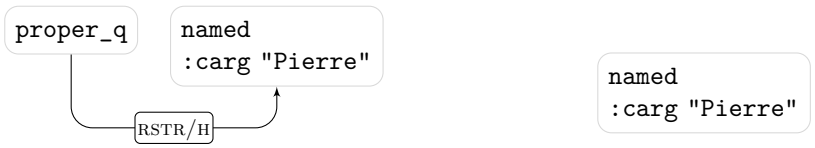
All nodes that are individuals are in the restriction of some quantifier. These can be informative, as with `_the_q`, `_any_q`, or `_neither_q` in the ERG, but they can also be simply the default or obligatory quantifier for a category of nodes. Examples of the latter are `udef_q`, the default quantifier for common nouns, and `pronoun_q`, the obligatory quantifier for pronouns in the ERG. Table 5.3 shows the default quantifiers for four kinds of nodes in the ERG and Jacy.

Generally these quantifiers can be removed without a loss of information, as re-adding

Category	ERG	Jacy
Common nouns	`udef_q`	`udef_q`
Pronouns	`pronoun_q`	`def_q`
Named entities	`proper_q`	`def_q`
Number names	`number_q`	–

Table 5.3: Default and obligatory quantifiers in the ERG and Jacy

them would be trivial given the predicate of the quantifiee. One exception is `number_q` in the ERG, which is used when talking about a number (e.g., *Two is the only even prime number*, or *Kim is in room 504*), but not when the number is used, e.g., for counting things or for times (*The two cats tumbled off the window sill.*, *I left work at 5:04pm*). Therefore, recovering the proper use of `number_q` would require more sophisticated graph analysis than just looking at the predicate of the quantifiee. Figs. 5.14a and 5.14b show the simplification of *Pierre* by removing its default quantifier.



(a) Original (b) Simplified

Figure 5.14: Removing implicit quantifiers

5.7.2 Combining Node Attributes

Predicates representing proper names, such as for person or organization names, days of the week, numbers, etc., use a generic predicate for their category, then the constant value

is associated via the node attribute CARG.[28] By embedding the constant value into the predicate name directly, the graph can be made simpler, as shown in Figs. 5.15a and 5.15b. A similar technique can be used for other node properties, such as the morphosemantic properties.

(a) Original (b) Simplified

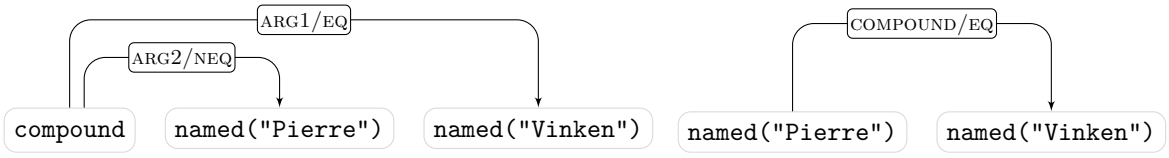
Figure 5.15: Simplifying predications with constant values

5.7.3 Converting Binary Predications to Links

There is a class of predications in MRS that take two arguments, where the one linked by the ARG1 role shares its label, and the other one is linked by the ARG2 role. In DMRS terms, this class contains nodes that are the source of two links: one ARG1/EQ and the other ARG2/NEQ. Examples of this class in the ERG include compounding, many prepositions (e.g., for *at*, *from*, etc.), and possessives, among other constructs. This class can be losslessly converted into (binary) links, as shown in Figs. 5.16a and 5.16b. This conversion might be best used selectively, as there may be some instances that are best left as full nodes, depending on the application (e.g., *fond* in *students fond of physics enrolled in the course*). I do not, however, convert binary predications to links in my experiments, as I consider the conversion, as defined, incomplete. There are also binary predications exhibiting the opposite pattern, where, in DMRS, a node is the source of ARG1/NEQ and ARG2/EQ links. Instead, I leave such transformations for future work, but I include the description here for completeness. Ivanova et al. (2012); Oepen et al. (2014, 2015) convert binary predications

[28] I call CARG a *pseudo-argument* as it is encoded in MRS with a role, but unlike other arguments it does not describe the EP's relation to some other EP. In DMRS it is just a node property, so in the visualization I represent this by including it in the node box.

when they are abstract (e.g., for `compound`, but not for `_at_p` or `_fond_a_of`), because otherwise the information would be lost in a bilexical representation. Yin et al. (2014) found that similar simplifications decreased data sparsity in a parse-ranking model and improved results.



(a) Original (b) Simplified

Figure 5.16: Converting binary predications to links

5.7.4 Summary

Using the transformations defined above, I can simplify the original example to the DMRS-like structure in Fig. 5.17. In my experiments, I make use of the first two transformations: removing nodes (including default quantifiers) and combining node attributes. I do not use the final transformation (converting binary predications to a links) at this time, but I leave it to future work. Copestake et al. (2016) describe further transformations, such as normalizing composed number names and proper names (e.g., such that *Pierre Vinken* is a single `named` node), but I do not attempt these transformations as they are more lossy than what I have described above, and would thus be harder to recover for generation.

Figure 5.17: Final simplified DMRS graph

5.8 Chapter Summary

In this chapter I have described several semantic operations for transforming, testing, and exploring DMRS graphs. In Section 5.1 I defined functions for converting MRS structures into DMRS, and in Section 5.2 I define the inverse procedure—converting DMRS back to MRS. In Section 5.3 I define three semantic tests: connectedness (Section 5.3.1), isomorphism (Section 5.3.2), and link orientation (Section 5.3.3). These tests are useful for detecting if DMRS structures are useful for bilingual alignment, and for serialization. In Section 5.4 I describe two methods of DMRS traversal. First, a surface-order node iteration (Section 5.4.1), then a traversal of the graph (Section 5.4.2) which uses the link orientation test from Section 5.3.3. I describe how I serialize DMRS into the compact PENMAN notation in Section 5.5. In Section 5.6 I cover two methods of extracting subgraphs. The first (Section 5.6.1) relies on a list of predicates to extract connected subgraphs with those predicates, and the second (Section 5.6.2) enumerates subgraphs of various traversal depths taking each node in the graph as a separate root. Finally, in Section 5.7 I show several simplification procedures of DMRS that are mostly lossless, including the removal of default quantifiers (Section 5.7.1), combined node attributes with the predicate (Section 5.7.2), and converting binary predications to links (Section 5.7.3). These operations are used heavily in bilingual subgraph alignment (Chapter 6), transfer rule creation (Chapter 7), and data exploration (Chapter 8).

Chapter 6

BILINGUAL SEMANTIC SUBGRAPH ALIGNMENT

The semantic operations defined in Chapter 5 enable me to describe processes of bilingual semantic subgraph alignment. I describe these processes assuming a bilingual MRS (bisem) corpus (see Sections 8.5 and 9.3). The goal is to identify source and target subgraph pairs $\langle g_s, g_t \rangle$ where each subgraph is connected and g_s and g_t are **bilingually correlated** (i.e., likely to be translationally equivalent). I hypothesize that non-trivial subgraph pairs, i.e., those leading to transfer rules of multiple EPs (see Chapter 7), have benefits for translation. Some words are lexically ambiguous, such as 手 *te* "hand/arm/technique/etc.", and having some context can help in selecting the proper translation, thus improving translation adequacy and fluency. Subgraphs also allow for non-compositional translations, such as idioms, where it's not easy to get a proper translation by transferring smaller portions separately.

I define two methods for finding alignments, each approaching the problem from a different angle. The first, described in Section 6.1, simplifies the semantic representation so it is feasible to use existing n-gram based word-aligners to find bilingually correlated semantic information. These correlations are then projected onto the original representations to recover the structural information. The second, described in Section 6.2, enumerates the possible subgraph pairings, with structure intact, of a $\langle g_s, g_t \rangle$ pair, then calculates statistics over the corpus to assign a score of bilingual correlation, which can then be used for filtering. Both of these kinds of subgraph pairs are used in Chapter 7 for the construction of MRS transfer rules. In Section 6.3 I discuss ways of filtering the extracted subgraph pairs in order to get rid of pairs unlikely to be useful. I compare my approach to related work in Section 6.4.

6.1 Bilingually Aligned Predicate Phrases

This method of subgraph alignment works by the proxy of aligning linearizations of graph nodes. Standard alignment techniques in machine translation work on n-grams of sentence tokens. These techniques do not readily apply to non-linear structures like DMRSs, which are directed acyclic graphs (DAGs), but if I can transform the DAGs into simple path graphs, then I can make use of the n-gram techniques.

This system uses Anymalign (Lardilleux et al., 2012), a bilingual word aligner, in order to find **predicate phrase pairs** from linearized predicate-only representations of the semantic graphs. Other word aligners, such as Giza++ (Och and Ney, 2003), may be used as long as they produce n-gram phrase alignments and not just word alignments.[1] These predicate phrase pairs have scores attached to them that indicate their bilingual correlation in the corpus. For those pairs whose score meets a threshold, I **project** the predicates onto matching semantic graphs (that is, I mark the nodes matching the predicates) in order to **extract** the graph structure that connects the predicates. The score from the aligner is then assigned to the extracted subgraph pair.

6.1.1 Surface-order Predicate Linearization

The input to Anymalign is a simplified linearization of the predicates in the semantic graphs. This linearization is produced using three operations from Chapter 5: surface-order node iteration (Section 5.4.1), node removal (Section 5.7.1), and combining node attributes (Section 5.7.2). As a running example in this section, consider the sentence fragment in (11)[2] and its original and simplified DMRS graphs in Fig. 6.1 for the Jacy-analyzed Japanese sentence[3] and Fig. 6.2 for the ERG-analyzed English sentence.

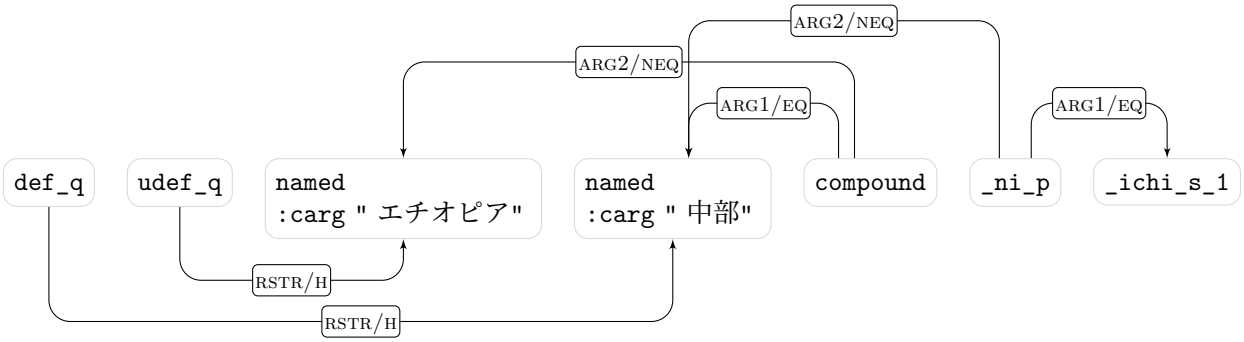
---

[1] My method expects the output format of Anymalign, so a converter or codec for other output formats would be required to make use of other word aligners.

[2] From the Japanese WordNet corpus; see Section 8.1.3.

[3] The given Jacy analysis is the first result from parsing, and 中部 *chuubu* "center" is analyzed as a name. Jacy also has a non-name predicate for the word, which is used in the second result (not shown), but otherwise the compounding construction is the same.

(11) エチオピア 中部 に 位置 する
Echiopia *chuubu* *ni* *ichi* *suru*
Ethiopia center LOC location do
"located in central Ethiopia" [jpn]



(a) Original

(b) Simplified

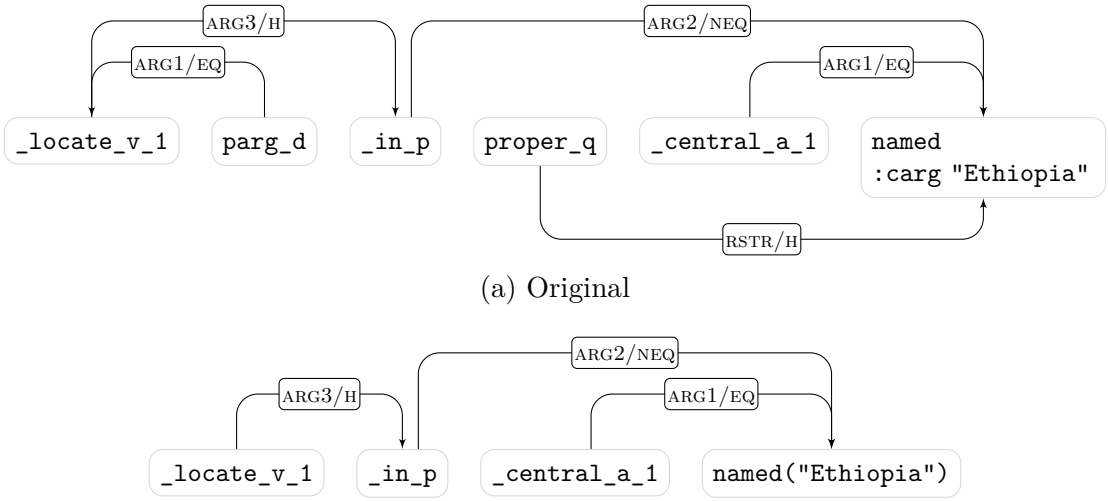
Figure 6.1: Original and simplified DMRS graphs for エチオピア中部に位置する

The first step is to iterate over the nodes.[4] The iteration order partially determines the final linearized form, but I do not linearize (i.e., join the predicate strings with spaces) yet as the following steps apply transformations that may utilize non-predicate node attributes.

In the next step, I remove specific nodes based on their predicates.[5] The n-gram aligner works best when tokens (predicates, in my case) that go together in a multi-gram alignment are adjacent within the string. Abstract predicates, particularly those that do not correspond to a content word in the sentence, can therefore interfere with the aligners ability to extract relevant predicate phrases. Some predicates I remove, such as quantifiers, I do not necessarily

[4]In the order as output from ACE, as discussed in Section 5.4.1.

[5]The actual predicates I remove are listed in Section 9.6.1.



(a) Original

(b) Simplified

Figure 6.2: Original and simplified DMRS graphs for *located in central Ethiopia*

want appearing in the resulting transfer rules, as I expect there should be rules in the hand-built grammar[6] for these closed-class predicates, or I can recover them as a post-processing step. Other predicates, such as for compounding or nominalization, I remove in order to help the aligner, but I allow them to be re-added during subgraph extraction (see Sections 6.1.2 and 9.6.1). This step may result in a disconnected graph, but for surface-order predicate linearization this is not a problem. In Fig. 6.1b, the nodes with predicates `def_q`, `udef_q`, and `compound` have been removed, while in Fig. 6.2b those with predicates `parg_d` and `proper_q` have been removed.

The next step is to combine node attributes. Constant argument values and morphosemantic properties, both attributes of nodes, are lost if I only linearize the predicate. Therefore I combine the constant argument value and properties with the predicate token. For example, if I encounter a `named` node with a `carg` property of `Kim`, the predicate token becomes `named("Kim")`. I similarly append some morphosemantic properties on pronouns (if they

[6]See Section 4.4.

are kept; in my experiments I remove them), since otherwise they all become normalized to the `pron` predicate in both the ERG and Jacy. For example, *she* in English becomes the token `pron(3.SG.F)`. While it's possible to do this for morphosemantic properties for most predicates, doing so would increase the data sparsity and be detrimental to the aligner's ability to find generalizable translations. In Fig. 6.1b the nodes with CARG values of エチオピア *Echiopia* "Ethiopia" and 中部 *chuubu* "center" have both been combined with their predicate to form `named("エチオピア")` and `named("中部")`, respectively, and in Fig. 6.2b the node with the CARG of *Ethiopia* has similarly been modified.

In the final step, I take the remaining nodes, extract the (possibly modified) predicate symbols, and join them with spaces to create the linearized predicate string. The predicates do not encode morphological variation, so this is similar to a lemmatized sentence. The result for Fig. 6.1b is `named("エチオピア") named("中部") _ni_p _ichi_s_1`, while for Fig. 6.2b it is `_locate_v_1 _in_p _central_a_1 named("Ethiopia")`. By doing this to both sides of the bisem corpus, I create a new kind of bitext (a *bipred* corpus, perhaps, by analogy of *bitext* and *bisem*) that will work with n-gram aligners like Anymalign.

6.1.2 Subgraph Extraction

I run Anymalign over the linearized predicate corpus to produce a set of predicate phrase alignments A, where each alignment $a \in A$ is a tuple $\langle S, T, W, P, freq \rangle$ such that S is a list of source predicates, T is a list of target predicates, W is a pair of lexical weights $\langle w_s, w_t \rangle$,[7] P is a pair of translation probabilities $\langle p_f, p_b \rangle$, and $freq$ is the frequency of the alignment. Once alignment is complete, I go over the bisem corpus again with the alignments in order to find aligned subgraph pairs. Recall from Section 5.6.1 that a single predicate phrase can match more than one subgraph for a DMRS instance, and this is true for both the source and target sides. Therefore, for every alignment, subgraph pairs are extracted if, for both the source and target predicate lists, there exists at least one connected subgraph in the correspond-

[7] W comes from Anymalign, but note that it is not used in Algorithm 9.

ing DMRS. The projection of predicates onto the DMRS and the extraction of connected subgraphs is accomplished by an extension of the DmrsSubgraphsFromPredList() function, defined in Section 5.6.1, called DmrsSubgraphsFromPredSpec(). This extended function takes an additional argument O, which is a set of optional predicates. Any optional predicate $o \in O$ is included in an extracted subgraph if all arguments of o are in the set of subgraph nodes N'. Algorithm 9 uses DmrsSubgraphsFromPredSpec() to define the extraction of subgraphs X from a corpus C using alignments A and optional predicates O (defined as a pair $\langle source, target \rangle$, as this is a bilingual function).

Algorithm 9 Extracting DMRS subgraphs using predicate alignments

Input: C is a bisem DMRS corpus as $\langle d_s, d_t \rangle$ tuples of DMRSs
Input: A is a set of predicate phrase alignments as $\langle S, T, W, P, freq \rangle$ tuples
Input: O is a pair of source/target optional predicate sets $\langle source, target \rangle$
Output: X is a list of aligned subgraphs as $\langle d'_s, d'_t, P, freq, count \rangle$ tuples

```
1 function ExtractSubgraphsFromAlignments(C, A, O)
2     for all ⟨d_s, d_t⟩ ∈ C do
3         for all ⟨S, T, W, P, freq⟩ ∈ A do
4             for all d'_s ∈ DmrsSubgraphsFromPredSpec(d_s, S, O^source) do
5                 for all d'_t ∈ DmrsSubgraphsFromPredSpec(d_t, T, O^target) do
6                     append ⟨d'_s, d'_t, P, freq⟩ to Y
7     for all unique elements ⟨d'_s, d'_t, P, freq⟩ ∈ Y do        ▷ Get extraction count
8         count ← number of times ⟨d'_s, d'_t, P, freq⟩ appears in Y
9         append ⟨d'_s, d'_t, P, freq, count⟩ to X
```

For the example in Figs. 6.1 and 6.2, Anymalign finds the alignment in Fig. 6.3. The procedure given in Algorithm 9 would only find one subgraph pair, shown in Fig. 6.4, from the DMRS pair in Figs. 6.1 and 6.2, as each predicate only has one matching node.

As discussed below in Section 6.4, the H&B rule templates not only match semantic material that can be transferred, but also encode structural constraints in the transfer rules and filter out bad or unlikely subgraphs, but the initial projection and subgraph extraction of my method only accomplishes the first of these: matching transferable semantic material. The semantic material at this stage is at the level of the subgraph, so all the mapping says

⟨(named(" エチオピア"), named(" 中部")),
(_central_a_1, named("Ethiopia")),
⟨0.767, 0.128⟩, ⟨0.226, 0.986⟩, 211⟩

Figure 6.3: Example alignment for エチオピア中部 → central Ethiopia

```
(x0 / named
    :carg "chuubu_3"
    :ARG1-EQ-of (e1 / compound
        :ARG2-NEQ (x2 / named
            :carg "echiopia_1")))
```

```
(x0 / named
    :carg "Ethiopia"
        :ARG1-EQ-of (e1 / _central_a_1))
```

Figure 6.4: Example subgraph pair extracted using the alignment in Fig. 6.3

is that the source subgraph can transfer to the target subgraph, but not, in the case of MWE pairings, which internal source nodes map to which internal target nodes. Therefore I need to subsequently perform filtering of the extracted subgraphs (Section 6.3) and apply structural constraints for both monolingual argument structure (Section 7.3.1) and bilingual mapping (Section 7.3.2).

6.2 Top-Down Subgraph Enumeration

This second method of subgraph alignment is a first step toward a method that is more native to nonlinear graphs. Rather than, as in Section 6.1, building a statistical model of bilingually correlated predicates and then extracting the structure for them, here I build a statistical model over the subgraphs directly. There are many ways this could be done: I could count occurrences over random subsets of subgraphs, akin to Anymalign for n-grams; optimize iteratively via expectation-maximization, as is done in Giza++ or synchronous tree grammars (Eisner, 2003; Ding and Palmer, 2005); or count enumerated subgraphs and build a one-shot statistical model over them to identify correlations. I choose to use the last of these methods, as it is easier to implement, although the others are fertile ground for future

work.

In Section 6.2.1 I explain how I enumerate and pair the source and target subgraphs. I describe the prefiltering of subgraph pairs in Section 6.2.2 as a preparatory step toward building a statistical model, which is done in Section 6.2.3. The weighted-ϕ^2 metric for filtering pairs using information from the statistical model is described in Section 6.2.4.

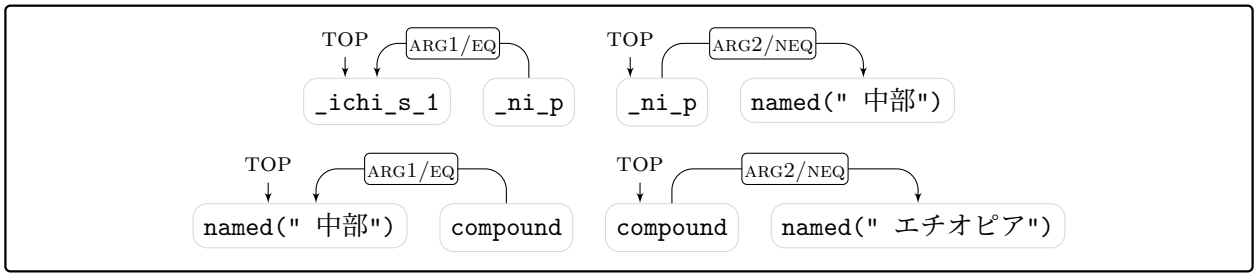
6.2.1 Enumerating and Pairing Subgraphs

I call this method *top-down* subgraph enumeration because I begin at the top of the graph, which for my data is always identified. The procedure for enumerating the subgraphs of a DMRS is an extension of the DmrsEnumerateSubgraphs() function, defined in Algorithm 8, called DmrsEnumerateSubgraphSpec(). The extended function takes an user-defined set of predicates, *drop*, such that any nodes in the traversal that have predicates in *drop* are excluded. As this may remove nodes in the middle of a traversal, any subgraphs that are disconnected will be excluded.

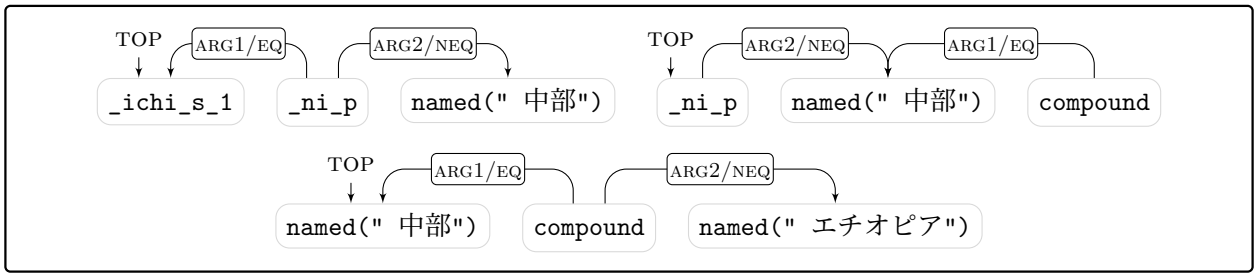
The subgraphs enumerated for Fig. 6.1a are shown in Fig. 6.5, and those for Fig. 6.2a are shown in Fig. 6.6. These subgraphs were created using different parameters than for the example in Section 6.1. In particular, nodes for the `compound` and `nominalization` predicates are kept. Note that the subgraph of the largest depth (four in Fig. 6.5 and three in Fig. 6.5) shows the full graph after simplification using the task-specifc parameters; those at all other depths are subgraphs of the largest one. Also note that I have rearranged the nodes from the original order. Since these subgraphs are created using a rooted traversal, I place the root (or top) at the left; all left-to-right links are regular *to*-oriented links, while right-to-left links are *from*-oriented.

Subgraph enumeration is a monolingual step, so the next step is to pair all subgraphs for a source DMRS to those of its target DMRS. Algorithm 10 defines how DmrsEnumerate-SubgraphSpec() is applied to both the source and target DMRSs to produce subgraphs which are paired, then counted, to create the intial model. Only subgraph pairs whose top nodes' types are the same will be paired. The counts will be used in the next steps to build

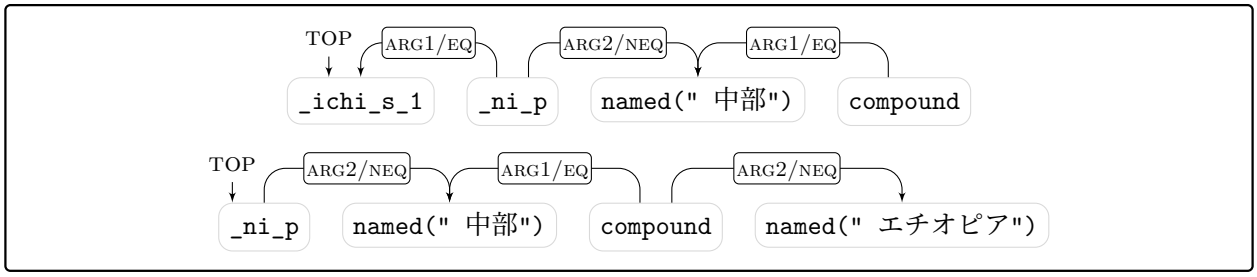
(a) Depth = 0



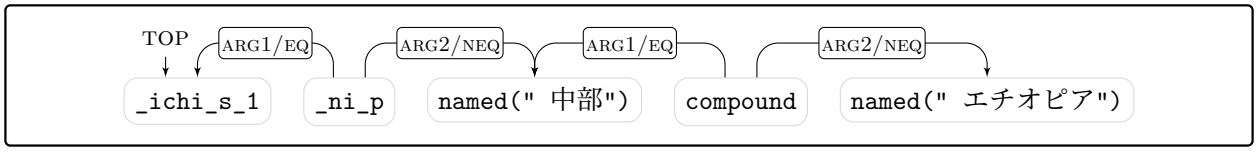
(b) Depth = 1



(c) Depth = 2



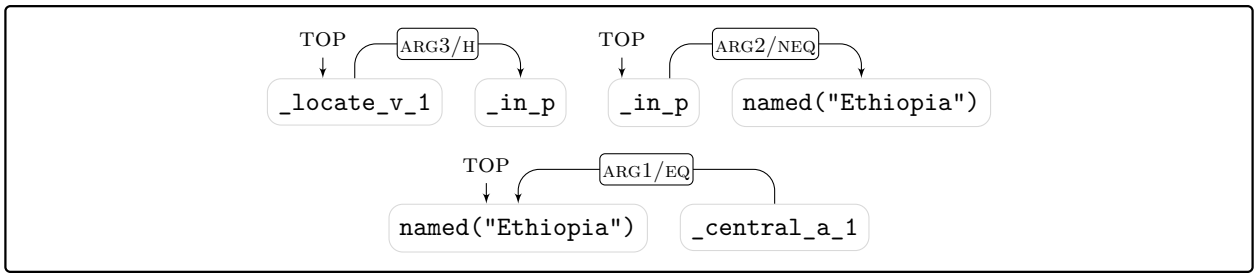
(d) Depth = 3



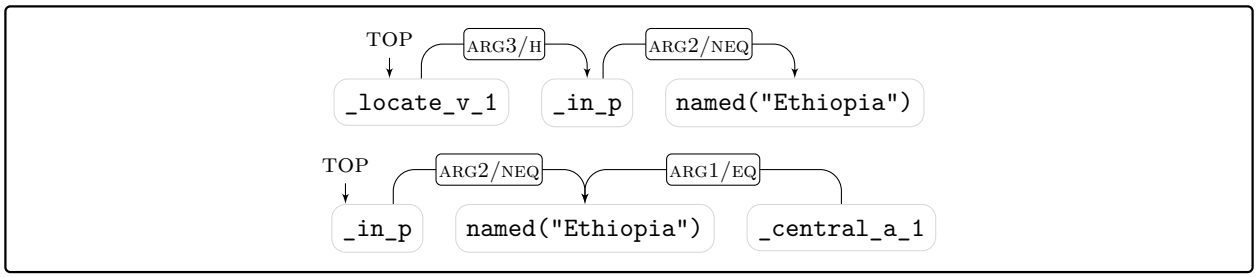
(e) Depth = 4

Figure 6.5: Example Japanese subgraphs enumerated for Fig. 6.1a

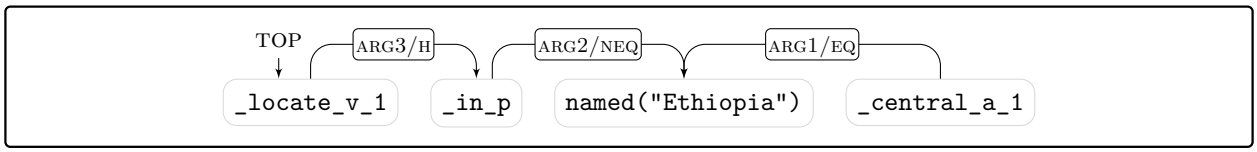
(a) Depth = 0



(b) Depth = 1



(c) Depth = 2



(d) Depth = 3

Figure 6.6: Example English subgraphs enumerated for Fig. 6.2a

the statistical model.

Algorithm 10 Pairing DMRS subgraphs

Input: C is a bisem DMRS corpus as $\langle d_s, d_t \rangle$ tuples of DMRSs
Input: md is the maximum depth of traversal
Input: $drop$ is a pair of source/target node sets to drop $\langle source, target \rangle$
Output: X is a mapping of subgraph pairs to counts $X_{s,t} = count$

```
1 function EnumeratePairs(C, md, drop)
2     for all ⟨d_s, d_t⟩ ∈ C do
3         for all d'_s ∈ DmrsEnumerateSubgraphSpec(d'_s, md, drop^source) do
4             for all d'_t ∈ DmrsEnumerateSubgraphSpec(d'_t, md, drop^target) do
5                 if VarSort(GraphTop(d'_s)) = VarSort(GraphTop(d'_t)) then
6                     append ⟨d'_s, d'_t⟩ to Y
7     for all unique pairs ⟨s, t⟩ ∈ Y do
8         count ← number of times ⟨d'_s, d'_t⟩ appears in Y
9         X_{s,t} ← count
```

6.2.2 Subgraph Prefilters

Before calculating statistics beyond simple counts, I perform prefiltering on the set of subgraph pairs that have been extracted. The model at this stage contains pairings of all enumerated subgraphs for each source and target DMRS pair, along with their occurrence counts. Noise in the model can drown out the signal, so I attempt to remove bad subgraphs before computing statistics. While I did some simple filtering in the enumeration step by excluding subgraphs beyond a depth limit and by excluding nodes in the *drop* sets, at this stage I filter out those that match a structural pattern. An example source and target pattern are given in Fig. 6.7, taken from the full list given in Table 9.9 of Section 9.7.1.

```
(e0 / unknown_v
```
(a) Source

```
(e* / _of_p)
```
(b) Target

Figure 6.7: Example prefilters

The source prefilter (Fig. 6.7a) specifies the node identifier as `e0` and its predicate as `unknown_v`, which will only match the top node of a subgraph if it has a matching predicate. The prefilter subgraph is "open" as there is no closing parenthesis, which allows it to match whether or not the node has any children. The purpose of this prefilter is to remove pairs where the source side's top node is an unknown eventuality, which occurs, for instance, when parsing just a noun phrase. The arguments of this unknown node in the original DMRS will, unless otherwise filtered, be enumerated separately, and thus can still be potentially used in transfer rules.

The target prefilter (Fig. 6.7b) specifies the node's type, but not its full identifier, so it will match whether or not it is the top node of a subgraph. It further specifies a predicate of `_of_p`, which in the ERG is used for *of* prepositional-phrases as in *the desert of Antarctica*, and the node is closed with a closing parenthesis, meaning that there are no incoming or outgoing arguments. The depth-based subgraph enumeration produces many subgraphs that capture only one or no arguments of nodes with binary arity, such as `_of_p`, `_and_c`, etc. Such subgraphs could be useful in translation (there could be many things that are *x of Antarctica*, e.g., *settlements*, *penguins*, etc., or for the other argument, *desert of x*), but this filter would exclude pairs where the target includes neither argument.

Prefilters are a graph-matching approach to removing pairs where one side exhibits an undesired pattern. If the prefilters are selected well, this step will remove noise from the model and make it easier for the next step to identify correlations.

6.2.3 Building a Statistical Model of Subgraph Pairs

In this step I calculate some statistics that can be used to filter uncorrelated subgraph pairs. Given that I have source and target subgraphs and a count, one of the easiest statistics to calculate are probabilities, such as the joint probability, forward and backward translation probabilities, and marginal probabilities for the source or target subgraphs. If X is the set of subgraph pairs and counts and $X_{s,t}$ is the count of a pairing of a source subgraph s and

target subgraph t,[8] then Table 6.1 defines some basic statistics using X.

Count of all observed subgraph pairs	N	$=$	$\sum_{s,t} X_{s,t}$
Count of all observed source subgraphs	X_s	$=$	$\sum_{t} X_{s,t}$
Count of all observed target subgraphs	X_t	$=$	$\sum_{s} X_{s,t}$
Joint probability	$P(s,t)$	$=$	$\frac{X_{s,t}}{N}$
Marginal source probability	$P(s)$	$=$	$\frac{X_s}{N}$
Marginal source probability	$P(t)$	$=$	$\frac{X_t}{N}$
Forward translation probability	$P(t\|s)$	$=$	$\frac{P(s,t)}{P(s)} = \frac{X_{s,t}}{X_s}$
Backward translation probability	$P(s\|t)$	$=$	$\frac{P(s,t)}{P(t)} = \frac{X_{s,t}}{X_t}$
Symmetric translation probability			$P(t\|s) * P(s\|t)$

Table 6.1: Basic statistics of enumerated subgraphs

Using just the forward translation probability for filtering is not very effective because it cannot easily distinguish correlation due to translational equivalence from other kinds of collocation. That is, for a translationally equivalent pair such as $s =$ `(e0 / _mitsukaru_v_1)` and $t =$ `(e0 / _find_v_1)`, $P(t|s)$ will be high because t occurs in may of the pairs where s does, but that is also true for a bad pairing, such as $s' =$ `(e0 / neg_x)` with the same t, simply because `neg_x` occurs frequently and thus pairs with many subgraphs. The same is true for the backward probability, but in the other direction. The backward translation probability could be used to distinguish the s and s' as $P(s|t) > P(s'|t)$, but it would suffer the same problem when the target t is generally frequent. The symmetric translation probability multiplies the forward and backward translation probabilities together, so a pair will only have a high symmetric translation probability if both the forward and backward probabilities are reasonably high. The symmetric translation probability is similar to pointwise mutual information, as shown in Eq. (6.1), and since $\forall_{s,t} X_s X_t \geq X_{s,t}^2 \geq X_{s,t}$, the difference

[8] Where $X_{s,t} = 0$ for any s and t where $\langle s,t \rangle \notin X$.

between the symmetric translation probability and pointwise mutual information is just one of scale.

$$\mathrm{PMI}(s,t) = \frac{X_{s,t}}{X_s X_t} \approx \frac{X_{s,t}^2}{X_s X_t} = P(t|s)P(s|t) \tag{6.1}$$

6.2.4 The Weighted ϕ^2 Filter

The translation probabilities defined in the previous section are all derived from (or are exactly) the conditional probabilities, which use the marginal probability in the denominator. The marginal probabilities, or similarly the count of source or target subgraphs, group together all occurrences of where the source or target appear, regardless if the other side (target or source) is included. In a contingency table, as shown in Table 6.2, the marginal probabilities or counts include the $\langle source, \neg target \rangle$ or $\langle \neg source, target \rangle$ pairs along with the $\langle source, target \rangle$ pairs. A more sophisticated metric would consider each cell separately.

	target	¬target	total
source	$a = X_{s,t}$	$b = \{X_{s,y} : y \neq t\}$	X_s
¬source	$c = \{X_{x,t} : x \neq s\}$	$d = \{X_{x,y} : x \neq s \text{ and } y \neq t\}$	$N - X_s$
total	X_t	$N - X_t$	N

Table 6.2: Bilingual contingency table

In feature selection for machine learning, the χ^2 method is often used for discovering which features have observed correlation different from the expected correlation, which is an indicator of the usefulness of the feature in a classifier. The translational equivalence of subgraphs is also a kind of correlation, so Church and Gale (1991) proposed using a variant based on the ϕ coefficient, defined in Eq. (6.2), for the discovery of translation pairs in

machine translation, as it is normalized by the corpus size and thus scales between 0 and 1.

$$\phi = \sqrt{\frac{\chi^2}{N}} \tag{6.2}$$

The definition of ϕ^2 in terms of the cells of the contingency table is given in Eq. (6.3). When more pairs appear in cells a and d ($\langle s, t\rangle$ and $\langle \neg s, \neg t\rangle$ pairs) than in cells b and c ($\langle s, \neg t\rangle$ and $\langle \neg s, t\rangle$ pairs), then s and t are correlated. When refactored and using my X-based counts, Eq. (6.3) is equivalent to Eq. (6.4). Fig. 6.8 shows the value of ϕ^2 for $a = 1$, $a = 5$, and $a = 25$—that is, when s and t are paired once, five times, or twenty-five times—for various values of the off-diagonal cells b and c with N constant at 10,000. Note that if there is only one observed pair $\langle s, t\rangle$, then introducing any $tuple \neg s, t$ or $\langle s, \neg t\rangle$ quickly drives down the ϕ^2 value. If the pair is observed twenty-five times, however, the correlation is robust to additional pairs involving s or t.

$$\phi^2 = \frac{(ad - bc)^2}{(a + b)(a + c)(b + d)(c + d)} \tag{6.3}$$

$$= \frac{(X_{s,t}N - X_s X_t)^2}{X_s X_t (N - X_s)(N - X_t)} \tag{6.4}$$

The ϕ^2 metric only considers the co-occurrence of subgraphs and not properties of the subgraphs, like graph order (number of nodes, or predicates, in the graph). This means that it will equally weight s and t whether they have an order ratio of 1:1 or 1:6. If the source and target subgraphs both contain predicates that occur only once in the corpus (e.g., a rare but translationally equivalent pair) then all enumerated pairings involving that predicate will get the same ϕ^2 value and, based on my ordering of transfer rules (see Section 7.2.2), the transfer grammar would always prefer the pairing with the largest source subgraph. I do not think this will always be the most accurate pair for translation, so I conjecture that, absent other information, translationally equivalent subgraphs should generally be the same order. I therefore weight the ϕ^2 value by the inverse of 1 plus the absolute difference in graph

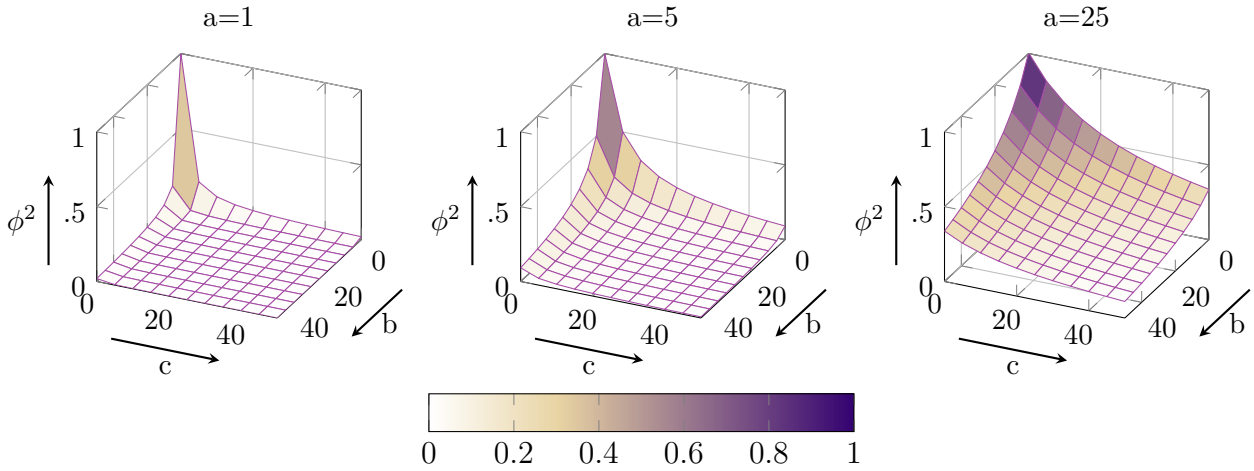


Figure 6.8: Example ϕ^2 values for various numbers of pairs (a), unpaired sources (b), and unpaired targets (c), in a sample size (N) of 10,000

orders, as defined in Eq. (6.5).[9]

$$W_{s,t} = \frac{1}{||V(s)| - |V(t)|| + 1} \tag{6.5}$$

For each pair $x \in X$ I calculate the forward and backward translation probabilities and the weighted ϕ^2 value. In the secondary filtering stage I exclude pairs that have a weighted ϕ^2 value below the threshold defined in Section 9.7.1. The translation probabilities are used in the final filtering stage, which is described in the next section.

6.3 Filtering Subgraphs and Subgraph Pairs

The two methods of discovering translationally equivalent subgraph pairs defined in Sections 6.1 and 6.2 both involve their own specific kinds of filters, but once the initial set of pairs (the *transfer pair store*) has been generated, there are additional filters that can be used to further reduce the set. Both methods assign a forward translation probability, a backward translation probability, and a count. As discussed in Section 6.2.3, neither the forward nor

[9]Note that the | character has two uses: in $|V(s)|$ and $|V(t)|$ it returns the order of the graphs s and t, then in $||V(s)| - |V(t)||$ it returns the absolute value of the difference of the orders.

backward translation probability are very useful for identifying translationally equivalent pairs, but the symmetric translation probability, which does a better job at identification, can be trivially calculated from these two probabilities.

I use the symmetric translation probability with thresholds defined in Sections 9.6.1 and 9.7.1 for filtering subgraph pairs. The threshold is not absolute. I treat the probability as a rank and select the top n pairs from each group of pairs with matching source subgraphs.

Monolingual grammars are often more robust in parsing than generation, and this robustness leads to many *unknown* nodes in the DMRSs. While it is possible to parse and transfer source-side unknowns, target-side unknowns are more problematic. As ACE is unable to realize sentences with unknown predicates, I filter out any pairs containing target-side unknowns.

6.4 Related Work

Jellinghaus (2007) automatically acquired MRS transfer rules for German (using the German grammar GG (Müller and Kasper, 2000; Crysmann, 2005)) and English (using the ERG (Flickinger, 2000)) using a top-down argument crawling method. The algorithm starts at the top of both the GG-produced and ERG-produced MRS instances and first creates *chains* of elementary predications by following explicit scopal edges, such as when two EPs share a label or when one is qeq to the other. The chains generally do not create a spanning path of the MRS, so Jellinghaus then, as much as possible, links together the disconnected chains after their initial creation. Next, the algorithm traverses scopal arguments, followed by non-scopal arguments, requiring that the values of the arguments are of the same variable type (e.g., `h`, `x`, `e`), and prefering arguments with the same role name. When there are alternatives in this traversal that are indistinguishable in the semantic graph, such as when two or more adjectives modify the same noun, the algorithm can make use of word alignments, as from Giza++, in order to resolve ties. These word alignments can also be used to help find EP alignments that are unreachable by argument crawling, or to provide validation for alignments that were found. Jellinghaus 2007, however, provides few details about how the

word alignment information is used in implementation, nor does he provide any evaluation that used these word alignments, so I cannot compare this part to my system. The argument crawling is similar, in some ways, to my top-down subgraph enumeration. The algorithm makes the assumption that the source and target semantic structures will be very similar; i.e., that their top predications are similar in meaning and that the predications have the same arity and same kinds of arguments. When this is not the case, the algorithm may fail to acquire transfer rules, or may possibly produce a rule that mixes up the arguments (such as with argument-switching divergences like *X likes Y* versus *Y pleases X*). This algorithm also emits transfer rules as it inspects each MRS pair and all rules are then collated so duplicates are removed and more specific (i.e., larger) rules appear before more general rules in the transfer grammar, but no statistics are collected for the purpose of weighting the rules. In contrast, my top-down subgraph enumeration does not treat scopal arguments as particularly special compared to non-scopal arguments. My method also does not a priori make the assumption that the source and target graphs will be structurally similar, although an isomorphism constraint may be enforced during filtering. Finally, I use rule weights for ordering transfer rules.

Haugereid and Bond (2011, 2012) described two methods for using n-gram aligners to align semantic subgraphs. In the first method, they aligned a tokenized and lemmatized bitext, then matched the lemmas to semantic predicates by looking up the lemmas in the lexicon. In the second method, they parsed the source and target sentence, linearized the predicate symbols, and aligned the linearization as though it were a bitext. In both methods, the result of alignment was a phrase table of aligned semantic-predicate n-grams. When both sides of an alignment matched one of the 22 transfer rule templates they defined, they generated a new transfer rule using the stored structure (i.e., semantic graph edges) from the template and the predicate content (i.e., the graph nodes) from the alignment. Weights from the phrase table are used to filter out unlikely transfer rules. The templates perform three roles: (1) in combination with the predicate alignments, find semantic material that can be transferred; (2) encode the transfer-rule constraints that link up the predicates monolingually

and map them bilingually; and (3) act as a filter for bad subgraphs (e.g., when the predicates match a disconnected subgraph). After extracting a large store of transfer rules, Haugereid and Bond later select rules that are applicable for a processing task by analyzing the MRSs to be transferred and filtering out rules that contain predicates that don't exist in the MRSs. The selection process can be easily repeated for subsequent translation tasks. My bilingually aligned predicate phrase method is similar to the second method of Haugereid and Bond (2011, 2012), but instead of matching the predicates to a template, I match them against the original semantic graph to extract a subgraph. I then create a transfer rule using the observed structure rather than using the structure encoded in a template. My method is thus more flexible and can match more kinds of structures, but it will also find more inappropriate pairings. I also adapt the rule-selection process for my pipeline.

Translation using deep syntax and semantics in LFG was described by Way (1999) as a variant of Data-Oriented Parsing (LFG-DOP) called Data-Oriented Translation (Hearne and Way, 2003). In LFG-DOT, syntax tree pairs from a bilingual corpus are decomposed into smaller tree fragments, which are then paired as a source of translation material. During translation, tree fragments are pulled from the *fragment base* and composed into larger structures. LFG-DOT's paired subtrees have much in common with my enumerated subgraph pairs. Being syntax trees, the fragments include non-terminal nodes which are used as the starting points of composition. In contrast, my subgraphs are semantic dependencies and do not have such non-terminal nodes. Machine translation using LFG's f-structures (Graham et al., 2009; Graham, 2011) is more similar to my method, as the *deep syntax* of f-structures is in some ways more similar to semantic structures than constituency trees.

In this dissertation I build on both the linearize-align-extract method of Haugereid and Bond 2011, 2012 and the traverse-enumerate-pair methods of Jellinghaus (2007), Hearne and Way (2003), Graham et al. (2009), and Graham (2011). I extend these methods by putting more focus on the graphical properties of the semantic representations and by exploring different filtering methods.

6.5 Chapter Summary

Using the semantic operations defined in Chapter 5, I have defined two methods for creating data stores of subgraph pairs for transfer. The first method, defined in Section 6.1, first finds translational equivalencies among n-grams of semantic predicates, then projects these onto matching semantic graphs to extract structural information. The second method, defined in Section 6.2, enumerates and pairs source and target subgraphs for each pair of DMRSs in the bisem corpus, then calculates stastics to build a model over the pairs. The two methods have their own specific kinds of filters to reduce unwanted pairings, then there are filters, described in Section 6.3, that apply to the outputs of both methods. Both methods extract pairs from the DMRS representations which, unlike syntax trees, are mostly lexical nodes—the hand-built core grammar has more general rules that target semantic constructions. I compare my methods to other related work in Section 6.4. The transfer pair stores created in this chapter contain DMRS subgraphs. In Chapter 7 I explain how I go from subgraph pairs to MRS transfer rules.

Chapter 7

TRANSFER GRAMMAR AUGMENTATION

The LOGON transfer machinery (Lønning et al., 2004) is a framework of tools and representations that is separate but related to that of parsing and generation. The aligned semantic subgraphs obtained by the method described in Chapter 6 are not in the form required by the LOGON transfer machinery, so in the next step I transform them into the type of rule the transfer machinery expects, called MRS transfer rules (MTRs). A **transfer grammar**, such as JaEn (Bond et al., 2011), the object of my research, is a system combining hand-written MTR types and some MTR instances, as well as a large set of automatically created MTR instances, built for the purpose of semantic transfer. As it is infeasible to manually create a large set of MTR instances, automatically extracting or generating MTRs is the standard way of giving the grammar reasonable coverage, first pioneered by Jellinghaus (2007) for rule extraction from source and target MRSs and by Nichols et al. (2007) for creating rules from bilingual dictionaries. This process of expanding the coverage of a transfer grammar is what I call **transfer grammar augmentation**—the subject of this chapter.

The MTRs produced by Haugereid and Bond (2011, 2012) are generated by inserting predicates into templates of hand-crafted MTR structures. In contrast, I extract MTRs from the actual semantic structure; any edges in the graph were observed in an MRS. One benefit of Haugereid and Bond's method is that it is easier to make use of the MTR types defined by the grammar, which can reduce redundancies among similar rules and generally simplify the form of the resulting rules. A downside to the template approach is that the templates can become out of sync with the source or target grammars. This can happen when the source or target grammars change in such a way that the MRSs they produce or

can generate from no longer match the input/output of the MTR templates. The templates can even become out of sync with the transfer grammar, e.g., by using an MTR type that has changed or has been removed. My method does not suffer from these downsides as it makes minimal assumptions about the core (hand-built) transfer grammar.

In Section 7.1 I give a quick walkthrough of the LOGON transfer machinery and the structure of MTRs. From a transfer pair store (created as described in Chapter 6), I first plan out which pairs will be used in an experimental grammar and how the resulting grammar will be structured (Section 7.2). I then describe the process of converting subgraphs to MTRs in Section 7.3, including the conversion of nodes and edges to feature structure representations, and the coindexation of variables to show subgraph-internal mappings. As the source and target grammars are likely to change over time, a transfer grammar should not be considered a static resource, so in Section 7.4 I describe issues related to the long-term maintenance of a transfer grammar.

7.1 The LOGON Transfer Machinery

The LOGON transfer paradigm (Lønning et al., 2004) builds on the previous Verb*mobil* conception of transfer as a resource-sensitive rewrite process of semantics (Wahlster, 2000), adding "(i) the use of typing for hierarchical organization of transfer rules and (ii) a chart-like treatment of transfer-level ambiguity." (Lønning et al., 2004, p. 3) Rule applications are successive and order-dependent; the input and output of most rules will not be the source or target representation, but some intermediate representation. My method for augmenting transfer grammars takes account of the chart-like processing of rules, in terms of rule ordering and optionality, but only minimally uses the transfer rule type hierarchy.

7.1.1 Anatomy of an MRS Transfer Rule

An MRS transfer rule has an identifier, a type, and four feature components: CONTEXT, INPUT, FILTER, and OUTPUT (Oepen, 2008). An identifier must be unique within a transfer grammar. The type is defined elsewhere in the transfer grammar and it may enforce

constraints that will be inherited by an MTR instance. The four feature components are optional and, if specified, describe a partial MRS. Most MTRs will have INPUT and OUTPUT specified. There is a fifth, relatively underdocumented component FLAGS,[1] which is used to signal the transfer engine to alter its default behavior when processing a rule. The optional rule type `monotonic_omtr` sets the feature FLAGS.OPTIONAL (see Fig. 7.3), but otherwise I don't directly use the FLAGS component in my rules.[2] The TDL (Krieger and Schäfer, 1994; Copestake, 2002) definition of an MTR is given in Fig. 7.1, and some examples are given later in Figs. 7.6 to 7.8.

```
mrs_transfer_rule :=  top &
[ FILTER mrs,
  CONTEXT mrs,
  INPUT mrs,
  OUTPUT mrs,
  FLAGS flags ].
```

Figure 7.1: TDL definition of `mrs_transfer_rule` in JaEn

An MTR will be applied if CONTEXT or INPUT match the input MRS and FILTER, if specified, does not match. Both CONTEXT and INPUT can be used to create bindings which are used to copy or constrain information between components, but only INPUT will consume the matched partial MRS (i.e., remove it from the MRS after matching). Any partial MRS descriptions on OUTPUT are inserted into the output MRS, with the bindings from CONTEXT or INPUT determining how it fits into the rest of the structure.

I only make use of two rule types: `monotonic_mtr` and `monotonic_omtr`. These types are shown in Fig. 7.2, as implemented in JaEn, which only binds the LTOP and INDEX features

[1] A short description is available at `http://moin.delph-in.net/LogonTransfer`.

[2] Other features include FLAGS.EQUAL and FLAGS.SUBSUME, which selects a method for comparing substructure, and FLAGS.BLOCK, which signals the transfer engine to ignore the current hypothesis. These features are used in some constructs in the core JaEn grammar, but are not directly utilized by my rules.

across the CONTEXT, INPUT, and OUTPUT components. The rules that I produce only use the INPUT and OUTPUT components, as the selection of constraints for the CONTEXT and FILTER components is a much more difficult problem, one my methods are not designed for. The `monotonic_omtr` type is for optional rules, which are discussed in Section 7.1.2.

```
monotonic_mtr := mrs_transfer_rule &
[ CONTEXT [ LTOP #h, INDEX #i ],
  INPUT [ LTOP #h, INDEX #i ],
  OUTPUT [ LTOP #h, INDEX #i ] ].

monotonic_omtr := monotonic_mtr & optional_mtr.
```

Figure 7.2: TDL definition of `monotonic_mtr` and `monotonic_omtr` in JaEn

7.1.2 Rule Optionality

When an MTR is applied, the elements on the INPUT component are consumed, and are thus no longer available to subsequent rules, but if it is not applied, it is skipped and the input MRS is passed on to the next rule. Inapplicable MTRs, i.e., those where CONTEXT or INPUT do not match the input MRS, or where FILTER does, will always be skipped. Applicable MTRs that are non-optional will always be applied. Applicable MTRs that are optional, i.e., those that have the FLAGS.OPTIONAL feature set, as shown in Fig. 7.3, will be both applied and skipped as separate hypotheses.

```
optional_mtr := mrs_transfer_rule &
[ FLAGS.OPTIONAL + ].
```

Figure 7.3: TDL definition of `optional_mtr` in JaEn

These forked hypotheses expand the search space and can, in theory, increase the like-

lihood of getting an acceptable target MRS, since the transfer process can attempt more paths. The expanded search space, however, increases the time and memory requirements of transfer, so it may lead to more timeouts or hitting the memory limit more often. Thus, the strategy for making rules optional should be sensitive to these limits.

My strategy is to group the rules by their INPUT and make all but the final rule optional. This ensures that all rules that are applicable for some partial input will be attempted. It does not, however, prevent larger rules from blocking smaller ones, nor does it prevent partially overlapping rules of the same size from being blocked. For instance, if the last (that is, non-optional) rule matching predicates A and B applies, then any rules matching just A or just B will be blocked, and so will rules matching, e.g., B and C, as B would have been consumed. I find the first situation, where larger rules block smaller ones, acceptable, if not ideal, because the expectation is that the larger rules transfer more idiomatic semantic fragments than multiple smaller rules, and are thus preferred. The second situation, rules blocking partially overlapping rules of the same size, is less acceptable, but here the expectation is that the tie-breaking criteria for rule ordering (see Section 7.2.2) will put statistically more likely rules before less likely ones of the same size.[3]

7.1.3 Summary

The LOGON transfer machinery is a powerful system for a resource-sensitive rewrite process to enable cross-lingual semantic transfer. I do not use all features of the system, but the ability to specify partial MRSs on the input and output with bilingual variable bindings and a strategy for expanding the search space via optional rules is enough to accommodate transfer from my extracted subgraph pairs.

---

[3]Meaning that within a set of rules of the same size, rules with the same INPUT will not necessarily be grouped together.

7.2 Challenges and Strategies in Building Transfer Grammars

Having a large number of extracted MTRs would seem to make for a more expressive and capable transfer grammar, but there are limits and tradeoffs to consider. Very large transfer models can cause the grammar to be too large to compile with ACE.[4] As of ACE `0.9.26`, the size limit of a compiled grammar is 2 GiB, which is space for approximately 120,000 extracted MTRs.[5] Also, as discussed in Section 7.1.2, larger models do not always lead to higher coverage or better results. The increased search space can cause the transfer process to hit time or memory limits more often, thus reducing coverage, and the inclusion of lower quality results can push the higher-quality ones past the result threshold.

Furthermore, timeouts in transfer do not only reduce coverage,[6] but they also make the experimental process take longer, which limits the capability to design and run numerous experiments. Assuming a timeout of 10 seconds and up to 5 parses to-be-transferred per input item, I estimate transfer of the Tanaka Corpus development set (4,500 items) would take $\frac{4{,}500*5*10}{3{,}600} = 62.5$ hours in the worst case where every input hits the timeout, but if only ∼10% hit the timeout and the rest take on average 100 ms, it would take $\frac{(450*5*10)+(4{,}050*5*0.1)}{3{,}600} = 6.8$ hours.[7] The actual time is somewhat less than this, as the source grammar does not have full coverage, and the average number of parses per (succesfully parsed) item is closer to four than five. Nevertheless, the ratio of items hitting the timeout to those that do not can have a large effect on the processing time.

---

[4] `http://sweaglesw.org/linguistics/ace/`

[5] 120,000 is an estimate based on the largest model I created, but the number will vary with the complexity of the extracted MTRs.

[6] This is practically but not strictly true. When ACE hits the timeout, it will output any hypothesis MRSs, whether they are fully or only partially transferred. When a timeout is reached, ACE spent its full time limit searching for transfers and never exhausted its search space, so it may in fact output many more MRSs than a search that does not hit the timeout. I do not count MRSs with untransferred material in the coverage, so a search that does not complete may only output MRSs that cannot be used.

[7] Processing can be parallelized, which speeds things up considerably. In my experiments, I parallelized about 20 jobs at once, but I also have around 20 configurations to process (see Sections 9.6.3 and 9.7.3), so for this time estimate the benefit is canceled out. Therefore, I ignore parallelization and multiple configurations in my estimate.

The two symptoms of large transfer models that I've identified—uncompilable grammars and inefficient grammars—require different solutions. For the first, I pre-select subgraph pairs relevant to the current task, while for the second I impose a rule ordering and selectively make MTRs optional in order to avoid placing excessive burden on the transfer engine.

7.2.1 Keeping Grammars Compilable with Task-specific MTR Selection

For the first symptom, uncompilability, I follow the tactic introduced by Haugereid and Bond (2012) where only the transfer pairs[8] relevant for a particular transfer task are used to build the model, and this pre-selection process is repeated for each task. MTR relevance is approximated by checking if the source predicates all exist in a semantic input, without considering the structure around those predicates. ACE also selects relevant MTRs prior to rule application, and the time required for this discovery process is subject to the timeout, meaning that pre-selection can also help the second symptom: transfer efficiency. ACE, however, is very quick to select MTRs from a compiled grammar, and the time is insignificant compared to the chart-based search of MTR applications. Pre-selection is therefore important mainly for maintaining the compilability of the transfer grammar, as the full transfer pair store could contain millions of subgraph pairs.

Taking this process to the limit—minimizing transfer grammar size by selecting relevant pairs for every input—would cause a significant increase in the time required to process each item. The time to select relevant pairs and compile a grammar is similar to that of the transfer timeout, so any gains made by having the smaller grammar would be dominated by the overhead of producing the grammar. Therefore I compromise, balancing transfer coverage and processing time, by building a transfer grammar customized for each test suite of 500 items. For some of my experimental systems, particularly with SGA (see Section 9.7), there were so many available pairs that even per-task pre-selection was not sufficient for keeping the grammar lean enough to be compilable, so I had to perform more aggressive

[8]Haugereid and Bond (2012) select the actual MTRs, but my selection process happens at the pre-MTR (i.e., subgraph pair) stage.

filtering at an earlier stage. Most configurations, however, were well under the 2 GiB limit for compilation.

7.2.2 Improving Transfer Efficiency with MTR Ordering and Optionality

For the second symptom, inefficiency, there are two treatments I apply. I first choose the order of MTRs within the transfer grammar. MTRs whose source MRS has more predicates (i.e., subgraphs of higher order) are placed before those with fewer, following the practices of Jellinghaus (2007) and Bond et al. (2011) who placed rules with more than one source predicate before those with one source predicate. For ties, I further sort by symmetric translation score (see Section 6.3), then the frequency, then the count.[9] This ordering makes MTRs that consume larger portions of the input more prominent, which has two intended effects: it prioritizes transfers that consider more source material at a time, potentially leading to more idiomatic translations; and it more quickly exhausts the input, which means transfer spends less time searching for rules to apply. The second treatment is the selective use of rule optionality as explained in Section 7.1.2, where each group of MTRs sharing the same INPUT are all made optional except for the last MTR in each group. In some preliminary runs, I attempted to make all of my extracted MTRs optional, which resulted in approximately 50% of the items hitting the timeout. With the rule-optionality strategy defined above, only about 5% hit the timeout.

7.2.3 Summary

My experiments, described in Chapter 9, extract a multitude of bilingual subgraph pairs, and without a careful strategy for selecting pairs and assembling them into a full transfer grammar, the grammar could be uncompilable or it could be inefficient. By pre-selecting rules that are relevant for the given set of inputs, I keep the grammar size small enough to compile.

---

[9] In the experimental system LPA (Section 9.6), the frequency is the number of times Anymalign (Lardilleux et al., 2012) found an alignment. Frequency is not relevant for SGA (Section 9.7). The count is the number of times my systems extract a transfer pair.

By ordering the rules by the number of predicates in the source subgraph, then setting all but the final MTRs for the same input as optional, I optimize the ability of the grammar to make use of its available rules without excessively increasing its computational requirements. Lastly, even though ACE does not employ statistical models for transfer ranking, the ordering of transfer rules creates a crude transfer model because rule application is ordered.

7.3 Converting Subgraphs to Transfer Rules

The semantic representation used in most stages of my transfer grammar augmentation process (extraction, filtering, and selection) is PENMAN-serialized DMRS subgraphs, but for compiling a transfer grammar these subgraph pairs must be converted to MTRs. The conversion process consists of three stages: monolingual structure conversion, bilingual variable binding, and MTR instance creation. The second stage, bilingual variable binding, is the most difficult and error prone, while the other stages are largely straightforward.

7.3.1 Monolingual Structure Conversion

In the first stage I must convert the source and target subgraphs to feature structures encoding the equivalent MRSs, retaining any argument structural and scopal relationships through monolingual variable binding. This conversion and variable binding is accomplished via DMRS-to-MRS conversion (see Section 5.2), using the first character of the PENMAN node identifier as the node's `cvarsort` (e.g., for `x1` it would be `x`). This conversion ensures that `*-EQ` edges (e.g., `ARG1-EQ`, `MOD-EQ`) in the subgraph result in label equalities in the MRS, that `*-H` edges (e.g., `ARG2-H`, `RSTR-H`) result in qeqs, and that regular arguments (e.g., `ARG1-NEQ`, `ARG1-EQ`) select the appropriate intrinsic variable of some other EP. The resulting MRS structures are not yet written to an MTR, but held in memory until after the variables have been mapped bilingually.

7.3.2 Bilingual Variable Binding

For the second stage I bind variables bilingually. While the monolingual binding encodes the relationship between EPs in a single MRS (i.e., a single side of the MTR), the bilingual binding identifies correspondences between the source and target MRSs. Without the bilingual binding, semantic material can still be transferred but it will be inaccessible to subsequent MTR applications, thus reducing the utility of the rule.

I bind the variables by building a bijection of source–target node identifiers where I have reasonable confidence of their correspondence. If the subgraphs are structurally isomorphic (see Section 5.3.2 for an explanation of structural isomorphism), I assume that each source node corresponds to the node in the respective position in the target subgraph. During the DMRS-to-MRS conversion of the subgraphs, I coindex (i.e., bind) the scope label, intrinsic variable, and any hole variables (for handle constraints) of the source and target EPs corresponding to nodes mapped in the bijection.

Fig. 7.4 shows an example subgraph pair for 恐ろしい夢を見る *osoroshii yume-wo miru* "have a terrible dream". The strings for structure comparison (shown in the top-right of the figure) are identical so the subgraphs are isomorphic, thus allowing a full bijection of node identifiers. The resulting MTR is shown at the bottom of Fig. 7.4 (MTR instance creation is explained in Section 7.3.3), and every source variable is coindexed with the corresponding target variable.

For non-isomorphic subgraphs, I assume they still have similar structure and I attempt to find an approximate mapping of node identifiers. Unlike the isomorphic mapping, this mapping will likely be incomplete, which means that more of the transferred material will be inaccessible to subsequent rules during transfer. For this mapping, I exploit the deterministic renaming of node identifiers to make the assumption that identifiers of the same form—that is, those with the same variable type[10] and traversal step—occupy equivalent positions in their respective subgraphs. This assumption (as with the assumption about isomorphism

[10]Recall from Section 5.5 that in the PENMAN serialization I embed the variable type into the node identifiers.

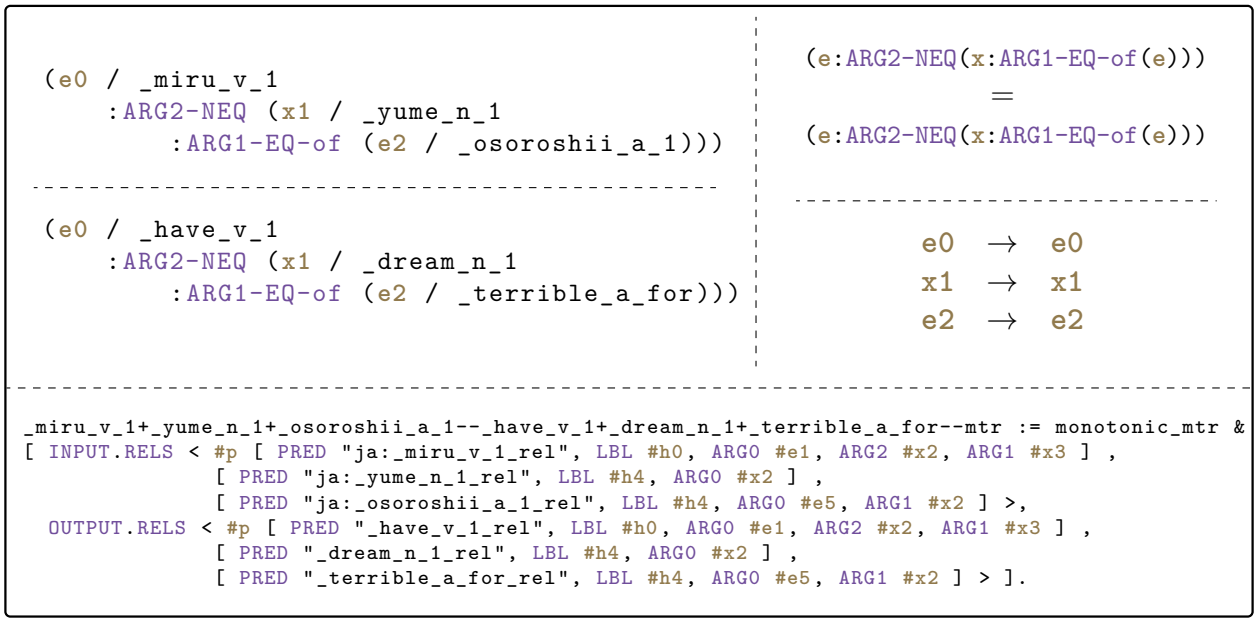


Figure 7.4: Source (top-left) and target (mid-left) subgraphs for 恐ろしい夢を見る *osoroshii yume-wo miru* "have a terrible dream", with the reduced form for structural comparison (top-right), the node-identifier bijection (mid-right), and the resulting MTR (bottom)

above) will sometimes be wrong, but I conjecture that it is better to have a partially correct bilingual binding than to have transferred material that cannot be expanded on at all. In order to get the bijection of node identifiers, I therefore take the intersection of source–target node identifier forms and use it to inform the binding of the scope labels and intrinsic variables of corresponding EPs. This method will always map the EP corresponding to the top subgraph node in the source subgraph (i.e., the first node in the PENMAN serialization) to the EP corresponding to the top subgraph node in the target subgraph because I only pair subgraphs with the same top-variable type (i.e., the top node identifier will always be `e0`, `x0`, `i0`, or `u0`). The non-isomorphic mapping may be incomplete, however, so I assume that any remaining nodes are unaligned. If I do not provide a mapping of variable forms for these unaligned EPs, their original form may unintentionally coindex with a mapped form or with one from another unaligned EP, so I add mappings for each unaligned variable

to a unique new variable. These unique mappings differentiate the variables for unaligned EPs so internal argument structure is retained without creating unintentional monolingual or bilingual correspondences.

Fig. 7.5 shows an example for すぐ医者に電話する *sugu isha-ni denwa suru* "call the doctor right away". These subgraphs are not structurally isomorphic, as shown by the structural comparision strings, so a full bijection of node identifiers is not guaranteed.[11] For the subgraphs in Fig. 7.5, only two of the four nodes for both source and target were mapped, but the MTR creation was able to integrate the other material by relying on the monolingual structure. For example, the Jacy predicate `_sugu_n_time` (for すぐ *sugu* "soon") modifies `_denwa_s_1` (for 電話 *denwa* "call") via the abstract predicate `unspec_p`, whereas the equivalent ERG predicate `_right+away_p` directly modifies `_call_v_1`. The nodes for these predicates were not mapped as they had differing node identifier forms, but both `unspec_p` and `_right+away_p` modify the appropriate EPs in the transfer rule (and both share a scope label with their modifiees) by relying on their respective monolingual structures which target nodes that were in the partial node-identifier bijection. Furthermore, the variables that were not bound from the partial bijection are differentiated (as `e4`, `x5`, and `h6` in the source side, and `h7`, `h8`, and `e9` in the target side) so they keep monolingual structure intact (e.g., `x5` links `unspec_p` and `_sugu_n_time`, and `h8` allows `_the_q` to qeq `_doctor_n_1`) without creating unintentional bilingual correspondences.

Both methods of bilingual variable binding (for isomorphic and non-isomorphic subgraphs) rely on many assumptions and are likely to introduce error, but without any binding the transfer process will suffer. Without the information about which source variables correspond to which target variables, the transfer engine can still replace one subgraph with the other and maintain monolingual argument structure. It will, however, be unable to link variables within the subgraph to those in the outer structure. Consider (12), where とても *totemo* "very" is an intensifier that attaches to うるさい *urusai* "noisy", and the structure

[11] It is possible to get a complete mapping if role names or structural attachment differ, as long as the subgraphs have the same count, order, and types of identifiers.

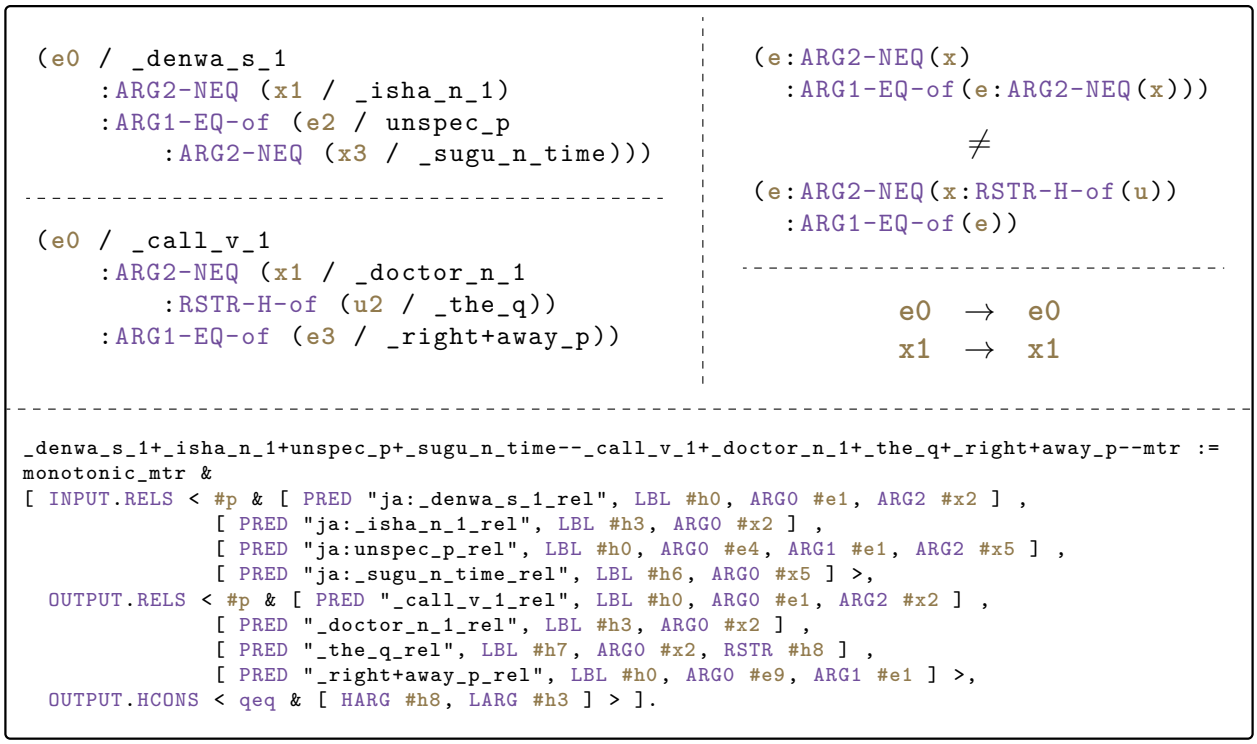


Figure 7.5: Source (top-left) and target (mid-left) subgraphs for すぐ医者に電話する *sugu isha-ni denwa suru* "call the doctor right away", with the reduced form for structural comparison (top-right), the node-identifier bijection (mid-right), and the resulting MTR (bottom)

is isomorphic in the source and target MRSs. If the transfer grammar has an MTR that accurately transfers うるさい犬 *urusai inu* "noisy dog" and a separate MTR that transfers とても *totemo* "very", then the target-side EP for *very* would only be able to attach to *noisy* if there was a bilingual variable binding for the うるさい *urusai* "noisy" EP on the source side and *noisy* EP on the target side. Otherwise, the EP for *very* would be unattached, leaving the MRS disconnected and likely unable to be realized as translations.

(12) とても うるさい 犬 が 吠える
totemo urusai inu -ga hoeru
very noisy dog -NOM bark
"The very noisy dog barks." [jpn]

7.3.3 MTR Instance Creation

In the final stage, I take the converted MRS structures and the bilingual variable bindings and produce an MTR instance. I first assign a valid MTR type—either `monotonic_mtr` or `monotonic_omtr`, depending on the relative location of the MTRs in the transfer grammar (see Section 7.1.2)—and a unique and valid identifier. I next add the INPUT and OUTPUT components to the MTR instance's attribute-value matrix, which will house the source and target MRS structures, respectively. I add the source and target EPs to INPUT.RELS and OUTPUT.RELS and handle constraints, if any, to INPUT.HCONS and OUTPUT.HCONS,[12] replacing the variables with those identified in the bilingual variable mapping (whether they are bound to the same form or differentiated). I also set a flag on the top EPs directly[13] for copying untransferred material within the EPs, shown in Figs. 7.4 to 7.8 as `#p` before the first source and target EPs. This flag is similar in form to variable coindexation, but serves a different purpose. It helps with the elements I do not model, such as variable properties,[14] and with role arguments that were not included of the original subgraph pair.

---

[12]Jacy currently does not have support for individual constraints, or ICONS, so this feature is not added.

[13]It is possible to flag other EPs in the same way, but I do not explore that possibility in this dissertation.

[14]The variable properties will then be transformed by the **variable-property mapping**, or VPM.

Consider the single-node subgraphs in Fig. 7.6, which represent the simplest possible pairing. Both nodes have `x` variables and single predicates without arguments, so the MTR in Fig. 7.6 only constructs EPs for the nodes with the scope label and intrinsic variable identified. Fig. 7.7 shows a pair of isomorphic subgraphs with two nodes each. The resulting MTR is able to bind all variables bilingually because they are structurally identical. Fig. 7.8 shows a non-isomorphic subgraph pair where the source has one node and the target has two. The non-isomorphic method of bilingual variable binding is, in this case, able to bind not only the scope label, but also the intrinsic variable of the source EP to the top EP of the target. The monolingual variable binding from the DMRS-to-MRS conversion of the target was able to coindex the ARG1 of the second target EP to the intrinsic variable of the first.

```
(x0 / _taiyou_n_1)                    (x0 / _sun_n_1)
```

```
_taiyou_n_1--_sun_n_1--mtr := monotonic_mtr &
[ INPUT.RELS < #p & [ PRED "ja:_taiyou_n_1_rel", LBL #h1, ARG0 #x2 ] >,
  OUTPUT.RELS < #p & [ PRED "_sun_n_1_rel", LBL #h1, ARG0 #x2 ] > ].
```

Figure 7.6: Subgraphs and MTR for 太陽 *taiyou* → *sun*

```
(x0 / _tegami_n                       (x0 / _letter_n_of
    :ARG1-EQ-of (e1 / _omoshiroi_a_1))    :ARG1-EQ-of (e1 / _interesting_a_for))
```

```
_tegami_n+_omoshiroi_a_1--_letter_n_of+_interesting_a_for--mtr := monotonic_mtr &
[ INPUT.RELS < #p & [ PRED "ja:_tegami_n_rel", LBL #h1, ARG0 #x2 ],
               [ PRED "ja:_omoshiroi_a_1_rel", LBL #h1, ARG0 #e3, ARG1 #x2 ] >,
  OUTPUT.RELS < #p & [ PRED "_letter_n_of_rel", LBL #h1, ARG0 #x2 ],
               [ PRED "_interesting_a_for_rel", LBL #h1, ARG0 #e3, ARG1 #x2 ] > ].
```

Figure 7.7: Subgraphs and MTR for 面白い手紙 *omoshiroi tegami* → *interesting letter*

MTR instance creation allows me to transform the subgraph representation I use for much

```
(x0 / _濁水_n_unknown)
```

```
(x0 / _water_n_1
    :ARG1-EQ-of (e1 / _murky_a_1))
```

```
_濁水_n_unknown--_water_n_1+_murky_a_1--mtr := monotonic_mtr &
[ INPUT.RELS < #p & [ PRED "ja:_濁水_n_unknown_rel", LBL #h5, ARG0 #x4 ] >,
  OUTPUT.RELS < #p & [ PRED "_water_n_1_rel", LBL #h5, ARG0 #x4 ],
                [ PRED "_murky_a_1_rel", LBL #h5, ARG0 #e7, ARG1 #x4 ] > ].
```

Figure 7.8: Subgraphs and MTR for 濁水 *dakusui* → *murky water*

of the transfer grammar augmentation process into the TDL-based representation required by the transfer machinery. It is the final step before a grammar is ready to be compiled and used for transfer.

7.4 Transfer Grammar Maintenance

As noted above, it is normal for a static transfer grammar to become stale, or no longer compatible, with respect to the source and target grammars it was created for. This can happen whenever the source or target grammars change their semantic model. For automatically augmented transfer grammars, like those explored in this dissertation, the stale MTRs can generally be refreshed to the modern grammars by rerunning the extraction process. It is therefore useful, when extracting MTRs, to note the version of the source and target grammars used, so that the maintainer or user of the transfer grammar knows when MTR refreshing might be needed.

The core transfer grammar can not be refreshed automatically. Rule types, hand-written rules, and any pre- or post-transfer transformations need to be updated manually. For a template-based approach like that of Haugereid and Bond 2011, 2012, the templates themselves can become stale, e.g., by targeting MTR types, source semantic patterns, or target semantic patterns that have changed. In these cases the template code must be manually updated.

The core JaEn grammar contains several thousand hand-built rules, but most of those are for proper names. In contrast, the automatically generated rules can number in the tens or hundreds of thousands (see Sections 9.6.3 and 9.7.3 for the rule counts of my experimental systems). Manually updating the much smaller set of hand-built rules is not infeasible and it is made easier by examining the logs of the generation step of the translation pipeline, which notes which predicates are incompatible with the target grammar.

7.5 Chapter Summary

In this chapter I have described the basics of the LOGON transfer machinery in Section 7.1. In Section 7.2 I explained how I overcame some challenges with producing efficient transfer grammars. In Section 7.3 I detailed how I create MRS transfer rules without templates Finally in Section 7.4 I discussed some issues of long-term transfer grammar maintenance.

This process of augmenting transfer grammars relies on the semantic operations defined in Chapter 5 and the methods of extracting bilingual subgraph pairs in Chapter 6. In Chapter 9 I make use of transfer grammar augmentation for a number of transfer grammar configurations.

Chapter 8
DATA EXPLORATION

I use three different data sources in my experiments, so in this chapter I will explore some properties of the data in order to better understand what my system is learning from. In Section 8.1 I list the sources with a brief qualitative description of their contents. I perform a basic analysis of the data in Section 8.2, and an analysis of parsing performance on the data in Section 8.3.

8.1 Data Sources and Divisions

This section briefly describes the three datasets. All three are Japanese-English sentence-aligned bilingual corpora publicly available under open licenses.

8.1.1 Tanaka Corpus

The Tanaka Corpus (Tanaka, 2001) is a modestly-sized corpus of roughly 150k sentences. The original corpus was produced by students of Professor Yasuhito Tanaka at Hyogo University. Professor Tanaka tasked each student with creating 300 translations, and after several years he had amassed 212k translations. This corpus, however, had translations taken from language textbooks, Bible verses, song lyrics, etc., and moreover there were a number of spelling and grammatical errors, duplicates, and mistranslations. It is unknown if the students translated originally Japanese sentences to English or vice versa, but it is likely that translations in both directions exist in the corpus. Revisions by subsequent maintainers of the corpus[1] removed bad or duplicated data and reduced the number to around 150k.

[1]See `http://www.edrdg.org/wiki/index.php/Tanaka_Corpus` for more information about the history of the corpus.

Sentences pairs are solitary translations, i.e., they are without any context from discourse. Tanaka (2001, pp. 1–2) claims that 40% of the sentences use a personal pronoun as the sentence subject, that the genre is "everyday-use sentences", and that ∼7.5% are interrogatives and ∼1% are exclamatives. Some examples from the corpus (glossing is my own, here and for examples throughout the chapter) are given in (13)–(15).

(13) 彼 は その 運動 に 関係 が ある の です か。
kare -wa sono undou -ni kankei -ga aru -no desu ka
3SG.M -TOP that campaign -DAT relation -NOM exist -NMLZ COP.HON Q
"Does he have anything to do with the campaign?" [jpn]

(14) その 単語 を 辞書 で ひい て ごらん。
sono tango -wo jiten -de hii -te -goran
that word -ACC dictionary -INS look.up -INF -try
"Look up the word in your dictionary." [jpn]

(15) 鈴木 さん に は 娘 が ３ 人 いる。
suzuki -san -ni -wa musume -ga 3 -nin iru
Suzuki -HON -DAT -TOP daughter -NOM 3 -NUMCL be
"Mr Suzuki has three daughters." [jpn]

The Tanaka Corpus is currently maintained by the Tatoeba Project,[2] but I use a version that is distributed with the Jacy Japanese grammar (Siegel et al., 2016) at `https://github.com/delph-in/jacy`. The corpus is closely linked with the grammar as it is one of the main sources of testing data used in the grammar's development. Jacy's version of the corpus contains some additional metadata about the well-formedness of each sentence.[3] For a small subset of the corpus, this metadata indicates the items are to be ignored because they, for instance, contain non-sentential data, multiple sentences in one item, etc. The other corpora do not have this metadata at all, so I elect to ignore the metadata and attempt to parse each item anyway.

[2] `https://tatoeba.org/`

[3] See `http://moin.delph-in.net/ItsdbReference` for an explanation of the `i-wf` field.

Francis Bond, the maintainer of the Jacy grammar, has defined splits of the Tanaka Corpus for development, testing, and training data. Equivalent amounts are set aside for development and testing data, then the remainder is for training. The sentence counts of each split are given in Table 8.1.

Development	4,500
Testing	4,500
Training	141,342

Table 8.1: Data splits for the Tanaka Corpus

8.1.2 Japanese-English Bilingual Corpus of Wikipedia's Kyoto Articles

The Japanese-English Bilingual Corpus of Wikipedia's Kyoto Articles (hereafter the *Kyoto Corpus*)[4] is a large collection of English translations of existing Japanese Wikipedia[5] articles about Kyoto. The translations were produced by Japan's National Institute of Information and Communications Technology (NICT)[6] for the purpose of aiding machine translation, so the translations are tight and sentence-by-sentence. The corpus contains nearly 500k translation pairs across 15 categories, such as *culture*, *literature*, and *railways*. The translations are close and accurate, but there are a large number of proper names, sometimes using the original Japanese orthography in the English sentences, and there are many long sentences. It, however, also has a number of very short entries, as everything from page titles to figure captions were translated. (16) and (17) are some examples from the Kyoto Corpus.

[4] `https://alaginrc.nict.go.jp/WikiCorpus/index_E.html`

[5] `https://www.wikipedia.org/`

[6] `https://nict.go.jp/`

(16) 1997 年 （平成 9 年） 5 月 22 日 京都 市営
1997 -nen (heiwa 9 -nen) 5 -gatsu 22 -nichi kyoto shiei
1997 -NUMCL (Heiwa 9 -NUMCL) 5 -NUMCL 22 -NUMCL Kyoto municipal
地下鉄 東西 線 の 開業 に 先立って 御池 駅 を
chikatetsu touzai -sen -no kaigyou -ni sakidatte oike -eki -wo
subway Touzai -line -GEN opening -DAT prior.to Oike -station -ACC
烏丸御池 駅 に 改称。
karasuma.oike -eki -ni kaishou
Karasuma.Oike -station -LOC renaming
"On May 22, 'Oike Station' was renamed 'Karasuma Oike' for the coming Kyoto Municipal Subway Tozai Line's opening." [jpn]

(17) 源氏 物語
genji monogatari
Genji story
"The Tale of Genji" [jpn]

The Kyoto Corpus did not have pre-defined data splits, so I split the data into contiguous blocks of development, testing, and training data following the order and proportions of the Tanaka Corpus. The sentence counts per split are given in Table 8.2.

Development	19,000
Testing	19,500
Training	448,730

Table 8.2: Data splits for the Kyoto Corpus

8.1.3 Japanese Wordnet Examples and Definitions

The Japanese WordNet (Bond et al., 2009) is a large semantic dictionary for Japanese, following the pattern of the Princeton WordNet (Fellbaum, 1998) for English. Both wordnets contain example sentences and definition sentences, so the Japanese WordNet Corpus[7] is the

[7]Available at `http://compling.hss.ntu.edu.sg/wnja/index.en.html`

pairing of example or definition sentences for entries that share a linked sense between the two wordnets. The pairing results in a corpus of about 180k sentences. Like the Tanaka Corpus, the sentences are relatively short, simple, and solitary. But, like the Kyoto Corpus, some entries are just a noun phrase or a single word, and are thus not full sentences. (18) is a sentence from the definitions subset, and (19) is a sentence from the examples subset.

(18) 太陽 の 大気圏 の 最も 外側 の 領域
taiyou -no taikiken -no mottomo sotogawa -no ryouiki
sun -GEN atmosphere -GEN SUPL outside -GEN region
"the outermost region of the sun's atmosphere" [jpn]

(19) 午後 は、 とても 前途有望 に 始まっ た
gogo -wa totemo zento-yuubou -ni hajimat -ta
afternoon -TOP very future-promising -ADV begin -PFV
"the afternoon had begun so promising" [jpn]

The Japanese WordNet Corpus, like the Kyoto Corpus, did not have pre-defined data splits, so I followed the pattern of the Tanaka Corpus. The sentence counts per split are given in Table 8.3.

Development	7,500
Testing	7,500
Training	168,968

Table 8.3: Data splits for the Japanese WordNet Corpus

8.1.4 Total Split Counts

The total sentence counts for all three corpora are given in Table 8.4. These sentence counts are for all sentence pairs, but not necessarily the number of items that my system uses in model training or in evaluation. See Section 8.5 for more information.

Development	31,000
Testing	31,500
Training	759,040

Table 8.4: Data splits for all corpora

8.2 Basic Analysis

I first count the number of sentences,[8] the vocabulary size, and the number of tokens in each corpus. In order to get the English tokens, the string is downcased and split on spaces. For the Japanese tokens, I use the MeCab morphological analzer (Kudo, 2005) to segment the string at each morpheme boundary. The counts for the full corpora are given in Table 8.5. The Tanaka Corpus and the Japanese WordNet Corpus have a similar number of sentences at 150,342 and 183,968, respectively, while the Kyoto Corpus has roughly three times as many at 487,230. The Kyoto Corpus has more than three times as many tokens, however, because the average sentence length is much longer. The standard deviation of sentence length is also much higher on the Kyoto Corpus, as there are many long and many short sentences, whereas the Tanaka and Japanese Wordnet corpora are more consistent in their sentence length.

Within each corpus, the token count for Japanese sentences is significantly higher than the English sentences, while the vocabulary size is significantly lower. This is because the Japanese tokens from the morphological analyzer splits morphological components from stems. Thus, each word in Japanese can result in multiple tokens, and because the morphology is fairly regular, these tokens often share the same form. The English data is not morphologically analyzed, nor is it stemmed, so each variation in form (e.g., *run*, *runs*, *ran*, *running*) counts as a single token but a different vocabulary item. To help avoid spurious

[8] I use the word *sentence* in this section to mean a translation entry, even if the data does not form a complete sentence.

variations, I lowercase all tokens when counting the vocabulary, but otherwise no normalization is done. Japanese, however, can represent the same word in multiple orthographies (e.g., 京都, きょうと, キョウト, or *Kyoto*) that cannot be easily normalized, so these can increase the vocabulary size.

			Tokens					
Corpus	Sentences	Vocab[a]	Total	Min[b]	Max	Avg	Stdev	Tok/Vcb
Tanaka-J	150,342	31,300	1,614,571	1	80	10.74	4.47	51.58
Tanaka-E	150,342	40,385	1,189,313	1	45	7.91	3.21	29.45
Kyoto-J	487,230	117,498	10,707,594	0	469	21.98	18.00	91.13
Kyoto-E	487,230	415,298	9,946,139	0	340	20.41	16.48	23.95
WordNet-J	183,968	46,923	1,897,032	1	106	10.31	6.71	40.43
WordNet-E	183,968	75,126	1,454,617	1	64	7.91	4.88	19.36
Total-J	821,540	137,501	14,219,197	0	469	17.31	15.42	103.41
Total-E	821,540	466,326	12,590,069	0	340	15.32	14.35	27.00

[a] English vocabulary is case-normalized
[b] Sentences of 0 tokens are blank lines; these are rare

Table 8.5: Sentence and token counts for the corpora

To get a better idea of the spread of sentence lengths across each corpus, I count the number of sentences for each token count, up to 70 tokens.[9] Fig. 8.1 plots this data with the purple bars for Japanese and the gold bars for English. This plot makes it clear that the Kyoto Corpus more evenly spreads its sentences across many sentence lengths, while the other two corpora cluster around the average.

Sentence length is a good predictor of the ability of a grammar to successfully parse a sentence. Table 8.6 lists the number of sentence pairs for each corpus where both the

[9] See Section 8.3 for the rationale for the 70-token maximum.

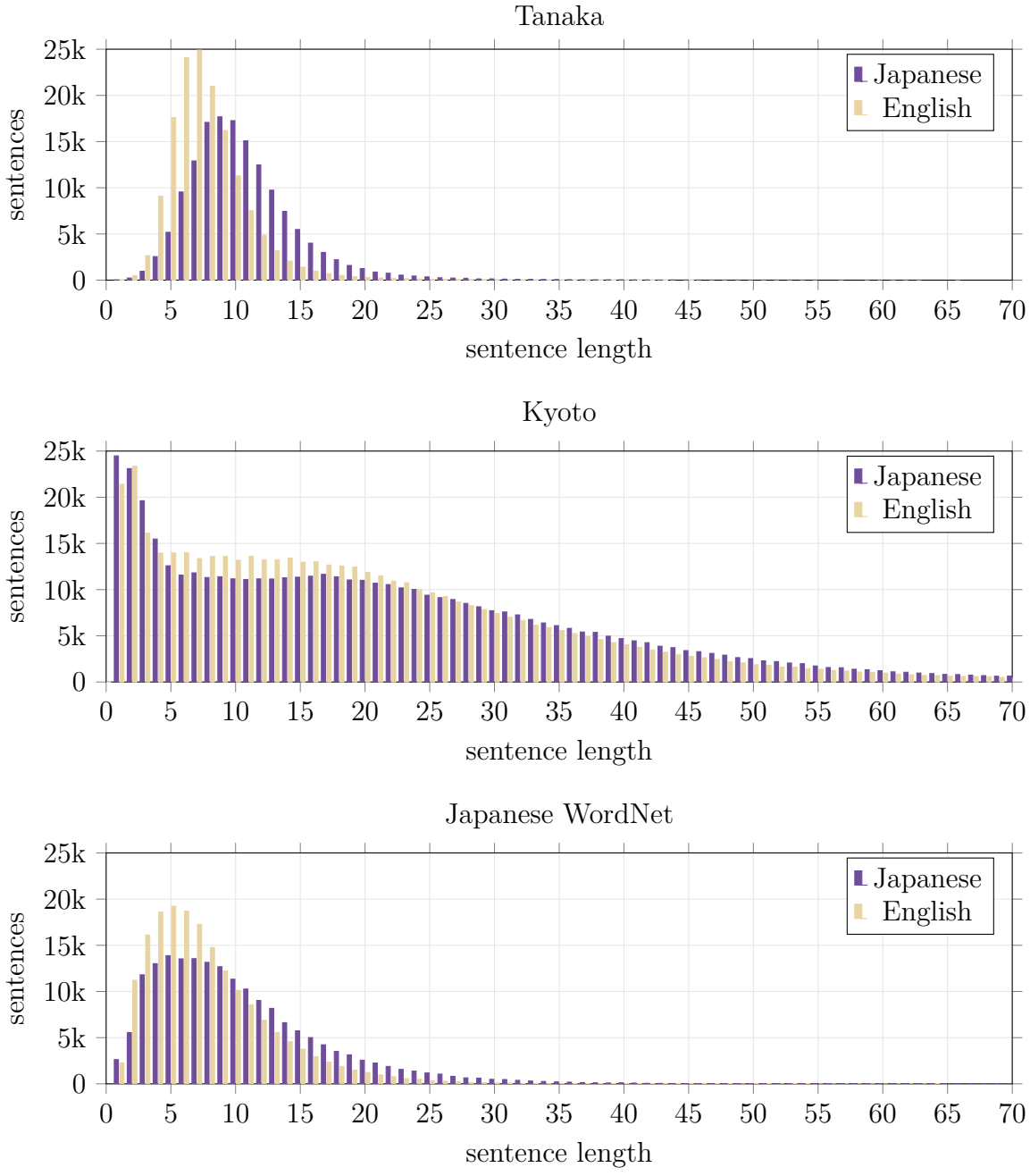


Figure 8.1: Number of sentences per token-count.

Japanese and the English side had fewer than or equal to 70, 35, or 20 tokens. As expected, there is little effect on the Tanaka or Japanese WordNet corpora, as they have very few long sentences, but the Kyoto Corpus is nearly halved when both sides have 20 or fewer tokens.

	Number of tokens		
Corpus	<= 70	<= 35	<= 20
Tanaka	150,341	150,002	145,547
Kyoto	476,375	378,549	244,925
WordNet	183,964	182,857	168,981
Total	810,680	711,408	559,453

Table 8.6: Number of sentences with a maximum of 70, 35, or 20 tokens

The corpora are also not free of duplicate entries, which can have a number of positive or negative effects on the system. Where it is useful to distinguish between duplicates that occur natural distribution of the data and those that occur because of workflow errors (e.g., where data is inadvertently copied or when splits overlap), I will call the former **identicals** and the latter **spurious duplicates**. For statistical systems that learn from patterns in the data, duplicates can increase the frequency, and thus the weight, of common patterns. In translation, one-sided duplicates (where one side, source or target, of a bilingual corpus is constant but the other side varies) can provide useful information about variability. In evaluation, a system might do better on test items that it has seen in training, and this is a situation where the difference between identicals and spurious duplicates is important. Leaking information (e.g., from spurious duplicates, intermediate results, or models) from training data in to the testing data is a kind of malpractice in data science that can cause artificially inflated evaluation scores, and sometimes it is not obvious where the leak has occured. When an item in one set (e.g., the training set) is duplicated in another (e.g., the

testing set), I will call it an **interset duplicate**, and when it occurs in the same set it is an **intraset duplicate**.

There are different schools of thought about how to handle duplicate entries. While it is generally the case that spurious interset duplicates should be avoided, practices differ on identicals and spurious intraset duplicates. One technique is to intentionally duplicate high-quality or important parts of the training data in order to give it more weight. For identical translation pairs (i.e., where the pairing of the source and target sentence are duplicated), one practice is to remove them throughout the corpus, especially for interset identicals. Others, however, consider it important to maintain the natural distribution of the data. If the testing data has identicals it performs well on, the evaluation score will go up compared to a fully deduped test set. Conversely, if the system does poorly on the identicals, the score will go down compared to the deduped test set. In these situations, the difference in score does not say much about the systems ability to perform well on different kinds of data, but it does say how well the system does on data that is representative of the natural distribution. Whether to remove duplicates, then, depends on the task at hand and the goal of evaluation. For this dissertation I choose to keep the natural distribution of the data and not remove any duplicates.

There are also more methodological considerations regarding duplicate translation pairs. If my system, for instance, noticed that a test item was identical to a training item and merely output a stored translation instead of doing the normal analyze-transfer-realize process, it might boost the evaluation scores but would simultaneously obfuscate the virtues of my methodology, and would thus be malpractice.[10]

Tables 8.7 to 8.9 show the duplicate counts across the training, development, and test data splits of the original corpus. There are three groups in the tables, one for duplicates only in the Japanese side, one for those in the English side, and one for those where the pair of ⟨Japanese, English⟩ sentences has been duplicated. Within each group, there are two

[10]For user-facing applications, however, this is a sound tactic that can increase reliable translations while reducing resource usage and response time.

columns: one for the number of original (i.e., first occurrence) items that have duplicates, and one for the number of redundant (i.e., beyond the first occurrence) items, such that the size of a deduped corpus is the redundant count subtracted from the original corpus size. The *total* rows in the tables are not the sums of the columns, but the duplicate counts considering all corpora together, so that repeated items across corpora are counted as well.

	Japanese		English		Translation	
Corpus	Original	Redundant	Original	Redundant	Original	Redundant
Tanaka	1	1	6,682	8,839	0	0
Kyoto	9,210	33,592	6,779	25,627	4,643	23,574
WordNet	1,514	2,043	1,668	2,391	849	1,235
Total	10,883	35,904	15,192	36,992	5,514	24,872

Table 8.7: Duplicate sentences in training data

	Japanese		English		Translation	
Corpus	Original	Redundant	Original	Redundant	Original	Redundant
Tanaka	0	0	14	15	0	0
Kyoto	314	738	254	582	179	484
WordNet	7	7	5	5	2	2
Total	322	747	273	602	181	486

Table 8.8: Duplicate sentences in development data

The Tanaka Corpus, having been revised to remove duplicates pairs, shows only one duplicate Japanese sentence in the training data, and none in the development or test data, and no duplicate translation pairs in any subset. There are, however, many duplicate English

	Japanese		English		Translation	
Corpus	Original	Redundant	Original	Redundant	Original	Redundant
Tanaka	0	0	9	9	0	0
Kyoto	343	965	256	793	211	706
WordNet	4	5	6	7	3	4
Total	349	972	271	809	214	710

Table 8.9: Duplicate sentences in test data

sentences, and the disparity between the number of Japanese and English duplicates is due to two or more Japanese sentences with very slight differences having the same English translation. (20) and (21) show two items with the same English translation but different Japanese sentences. The only difference is that (21) uses the third-person pronoun 彼 *kare* "he" instead of the second-person 君 *kimi* "you".[11] The other two corpora have not been revised to remove duplicates, and thus there are much higher numbers of them. The Kyoto Corpus has many repeated headings, such as *Summary*, which occurs 2,161 times in the training data. The Japanese WordNet Corpus has many duplicates across the Japanese and English sentences and the translation pairs. Compared to the Tanaka Corpus, which is similar in size, the Japanese WordNet Corpus has significantly fewer English duplicates because it is not plagued by the same problem of having slight differences in Japanese map to the same English sentences.

(20) 君 は 自分 で 重要 だ と 思う 本 を 読む べき だ。
kimi -wa jibun -de juuyou da to omou hon -wo yomu beki da
2SG -TOP REFL -INS important COP COMP think book -ACC read DEO COP
"You should read such books as you consider important." [jpn]

[11]Note that the translation of (21) is thus inaccurate—*He should...* would be better. The students who produced the corpus items may have created these kinds of near-duplicates by sharing and modifying translations with each other, or by using the same reference material.

(21) 彼 は 自分 で 重要 だ と 思う 本 を 読む べき だ。
kare -wa jibun -de juuyou da to omou hon -wo yomu beki da
3SG.M -TOP REFL -INS important COP COMP think book -ACC read DEO COP
"You should read such books as you consider important." [jpn]

Table 8.10 shows the number of test items that are duplicated in the combined training and development data. That is, the *original* columns show the number of unique test sentences that have duplicates in the development and training data, and the *redundant* columns show the number of times those sentences occur outside of the test set.

	Japanese		English		Translation	
Corpus	Original	Redundant	Original	Redundant	Original	Redundant
Tanaka	0	0	273	764	0	0
Kyoto	1,194	16,295	690	12,178	464	5,517
WordNet	143	262	163	362	82	167
Total	1,347	16,610	1,355	13,334	548	5,699

Table 8.10: Duplicates of testing data in the training and development data

8.3 Analysis of Parsing Performance

In this section I analyze the performance of a parser on the corpora. As my system requires semantic representations for both the source and target sides of the bitext corpora in order to extract transfer rules, the parsing results can show the upper bound of the training data that is usable, and can shed some light on the quality of the analyses.

I parsed the Japanese data with Jacy and the English data with the ERG using the ACE processor.[12] ACE only attempts to parse up to 70 tokens in an input item, so in the following charts I only show the results for up to 70-token sentences. Furthermore, I added additional

[12] `http://sweaglesw.org/linguistics/ace/`

constraints on the time and memory allowed per input item, as shown in Table 8.11. While these constraints can reduce the number of items that get a parse, they are beneficial in that they prevent very difficult items from taking excessively long to process, and they help to regularize the processing performance across multiple runs. I also split the parsing task into jobs of 500–1,500 sentences and ran the jobs on a computing cluster. Information about the cluster machines is given in Table 8.12.

maximum sentence length	70	tokens
maximum number of results	5	results
parsing timeout	10	seconds
chart memory	4096	MiB
unpacking memory	4096	MiB

Table 8.11: Parsing constraints for ACE

Operating System	CentOS 7
Processor	Intel Xeon 2.2GHz–3.5GHz
Memory	12 GiB allocated per job

Table 8.12: Machine information

The parsing coverage is plotted in Fig. 8.2. As before, Japanese results are in purple and English in gold. For ease of reference, the number of sentences per sentence length is plotted in light purple and light gold. Note that the coverage plots use the right axis and range from 0% to 100%, so, for example, Jacy has about 80% coverage of the ∼17k sentences of the Tanaka corpus with 10 tokens. In general the ERG, a more mature and well-rounded grammar, gets better parsing coverage than Jacy. All three corpora show

reasonably expected downward trends as the sentence length increases, although Jacy on the Kyoto Corpus has a steep decline until sentences of 6 or 7 tokens, then it levels to a more gradual decline. This trend, I suspect, is due to Jacy being less capable than the ERG of dealing with incomplete sentences, as the shorter sentences in the Kyoto Corpus are often titles, captions, headings, and other kinds of non-sentential material. After 35 tokens, the parsing coverage declines rapidly, and in the case of the Tanaka and Japanese WordNet corpora, the plots become erratic as there are so few sentences at the higher lengths.

Figure Fig. 8.3 plots at each sentence length the average time taken for a successful parse, with the upper-bound of the y-axis set to the timeout value of 10 seconds. As before, the sentence-length distribution, and now also parsing coverage, are plotted in a lighter color. I do not include timing information for failed parses as I do not have per-item information about those that failed due to exceeding the timeout. The ERG generally takes much longer than Jacy to reach an analysis, perhaps because it has a greater range of possible analyses to attempt, i.e., Jacy fails more quickly as it exhausts its search space. With the ERG, there is a correlation between the increase of parsing time and the decrease of coverage, suggesting that many of the items failed due to reaching the timeout. Failing due to timeout does not imply that a larger timeout would yield any results, as it could simply be an item the grammar cannot fully model. This latter case may be more applicable with Jacy here, as Jacy's parsing time is less correlated to coverage than it is with the ERG.

Fig. 8.4 plots at each sentence length the average memory required to parse an item, with the upper-bound set to 4 GiB, the maximum memory I allowed. As before, the sentence-length distribution, and now also parsing coverage, are plotted in a lighter color. The memory usage, as expected, gradually climbs with the increase in sentence size. While the memory usage of failed items is not included in these charts, the average usage is far from the maximum, and I only find one item failing due to the memory limit.

To summarize, the ERG exhibits greater coverage than Jacy, especially for corpora that are not the Tanaka corpus, but this coverage has a cost in terms of the time and memory required to get a parse. While English items that fail to parse with the ERG seem to failing

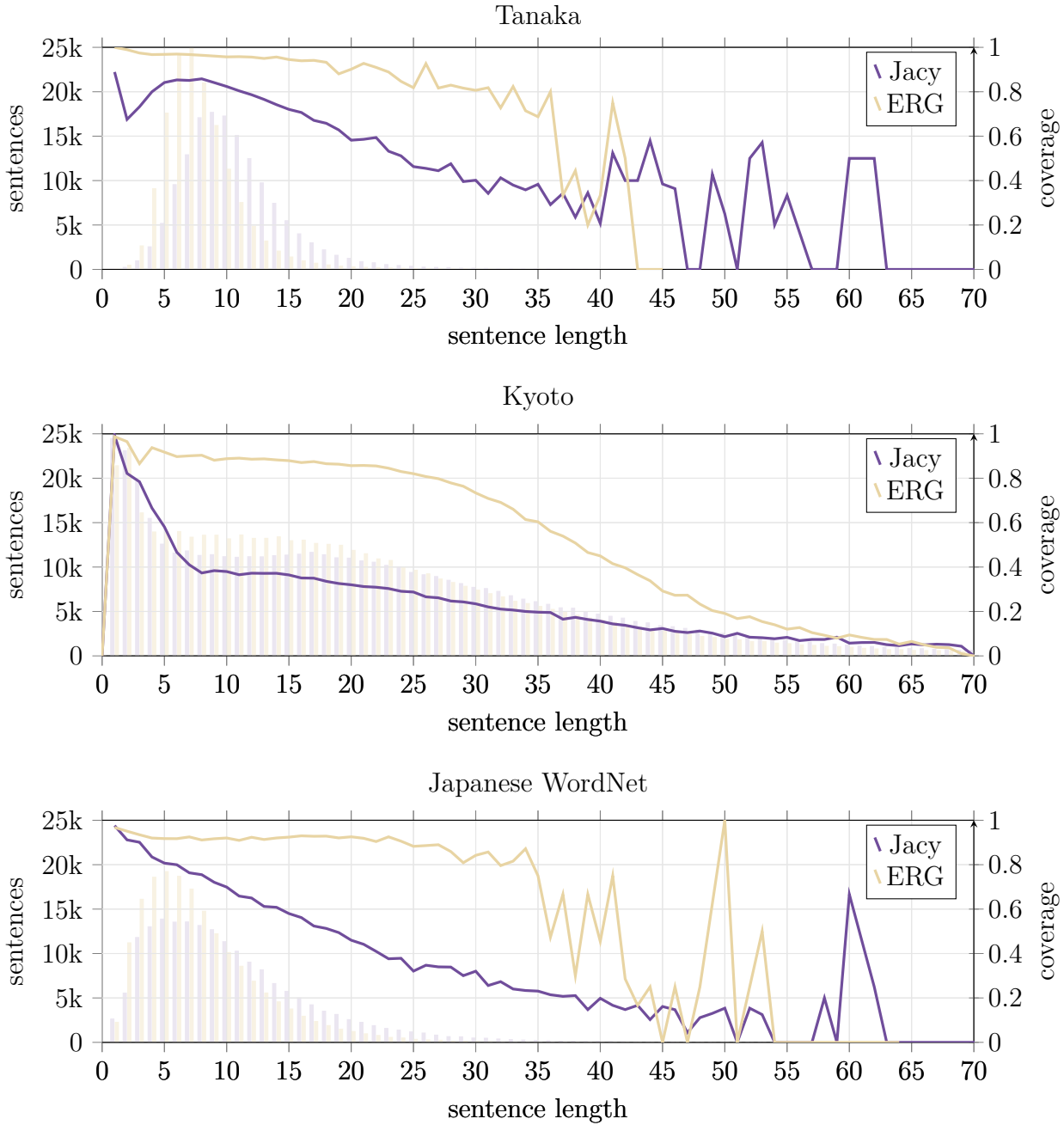


Figure 8.2: Parsing coverage for each sentence length.

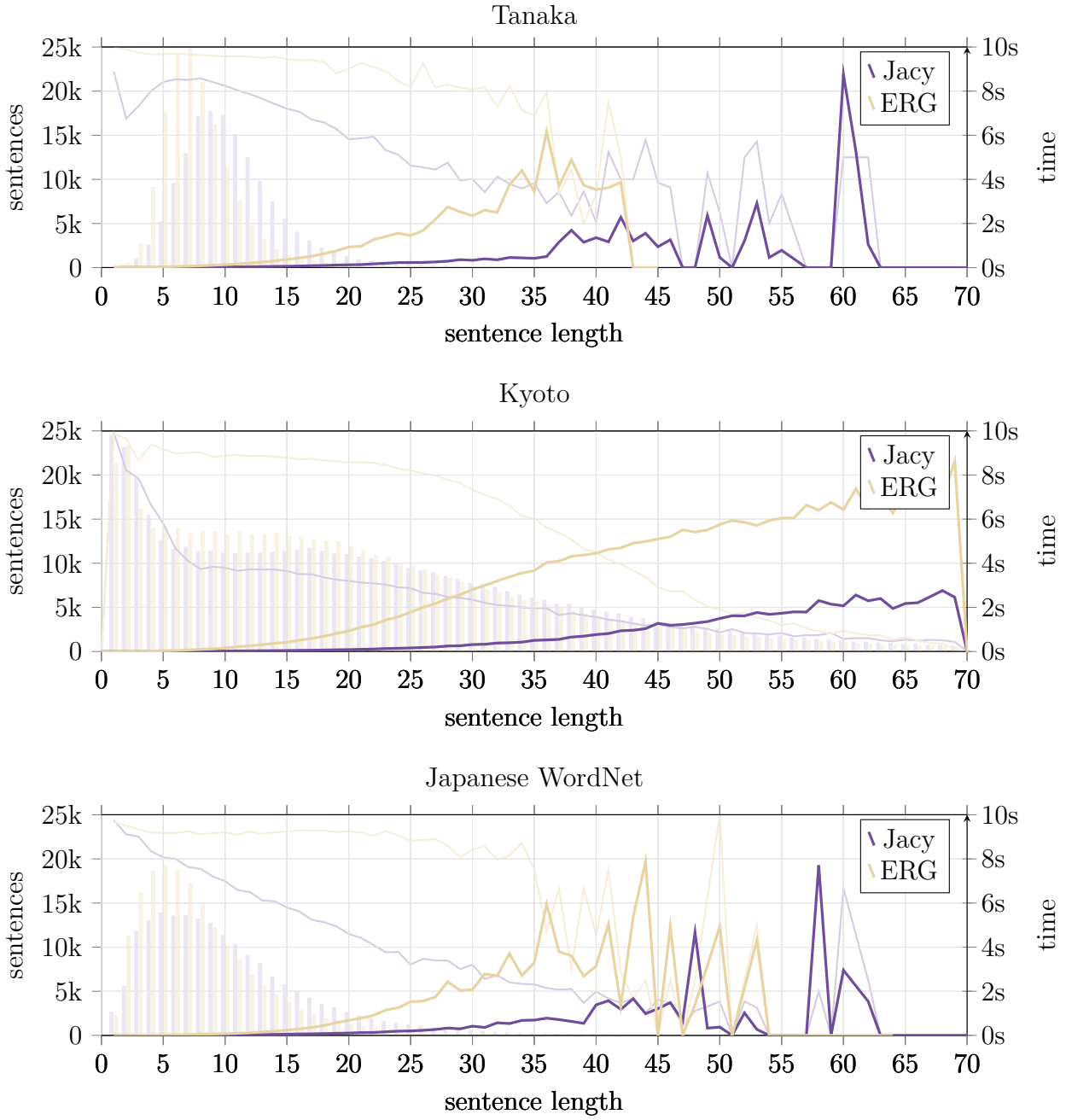


Figure 8.3: Average per-item parsing time for each sentence length.

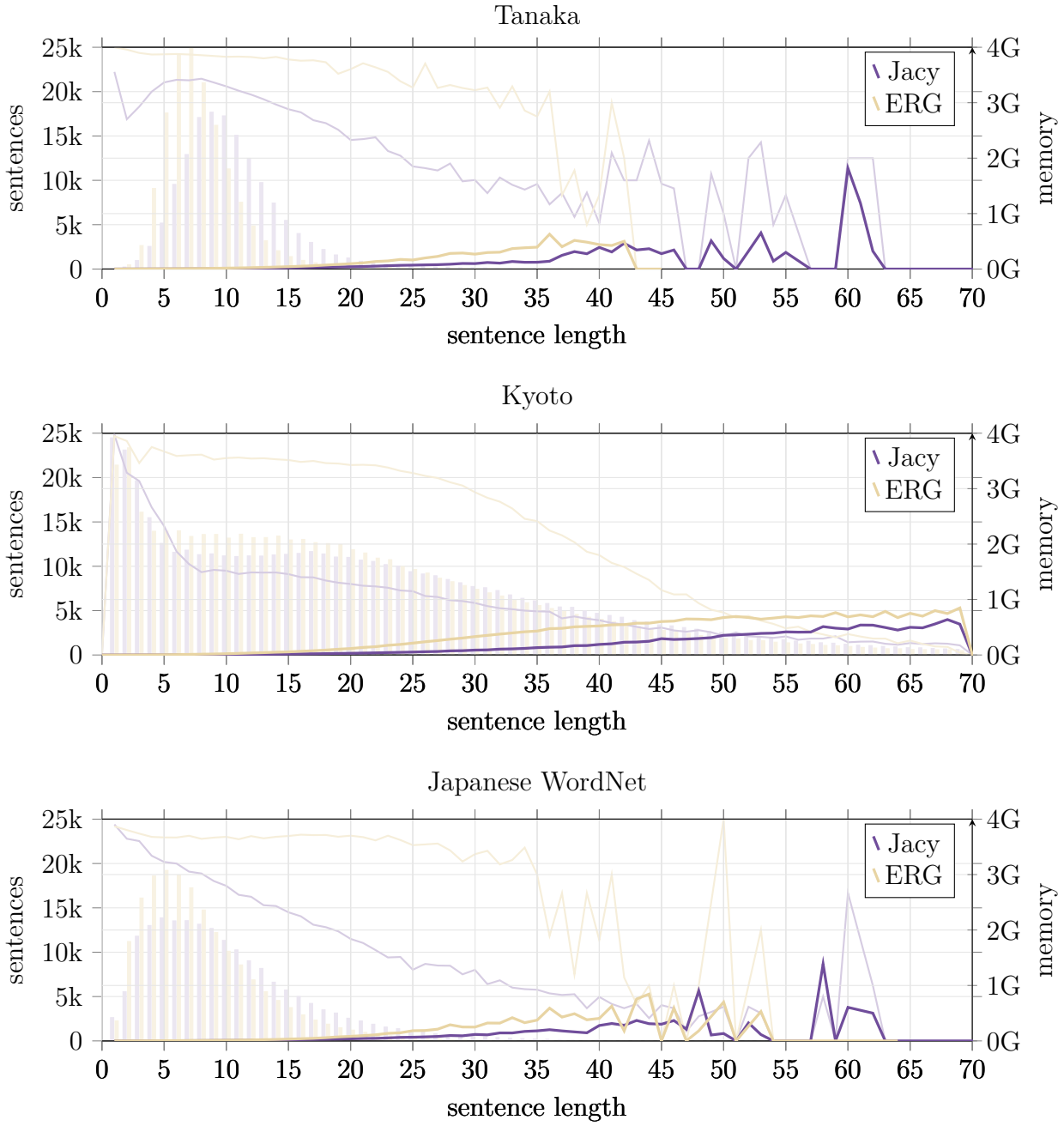


Figure 8.4: Average per-item parsing memory usage each sentence length.

due to the timeout, Japanese items do not show such a trend, and instead seem to failing simply due to a lack of encoded analyses or lexical entries.

8.4 Analysis of Generation Performance

The fact that the implemented grammars are bidirectional is important for my translation methodology, as I use the ERG to generate surface realizations from the semantic representations transferred from the Jacy-produced representations, but I can also use this functionality to generate surface realizations from ERG-parsed representations. This task is called *rephrasing* (also, *paraphrasing*), as opposed to *translation*. Rephrasing can be used to estimate the expected upper bound of translation performance, as it shows what the target grammar can generate given ideal semantic inputs. However, this is only an estimate because, even with ideal semantic inputs, the target grammar will not exhibit perfect performance, as described below. Also, it is possible to translate an item where the target grammar cannot parse the reference string (i.e., when the semantics transferred from the source representation **can** be realized by the target grammar), so rephrasing will not be able to show the target grammar's performance on those items at all.

The grammar can tell me every semantic representation it can assign to a sentence, assuming it does not time out or run out of memory during parsing, and these representations are also the full set that can generate to the original string, so this analysis task seems at first to be uninteresting—I should always be able to get the original string. There are, however, a number of reasons why it's not always possible to generate the originally parsed string. First, I am not storing every realization result, and, as discussed in Section 4.6, the realization-selection model is not as sophisticated as the parse-selection model, so it is possible that the top N realizations per input do not include the original sentence, even if it is possible for the grammar to generate it. Second, it's possible that the grammar is incapable of generating a sentence that it can parse (the core grammar itself is bidirectional, but the preprocessing and postprocessing steps may not be). For instance, the ERG will parse contractions like *they'll*, but its default configuration for generation is to only realize the full form (e.g., *they*

will), in order to cut down on uninformative ambiguity. Similarly, spelling variations will be normalized (e.g., *colour*/*color*), as will capitalization and punctuation. The ERG has unknown word handling so it can guess the syntactic function of words not in its lexicon, but ACE does not have the ability to generate from unknown word predicates.[13]

I analyze the ERG's generation performance of the English development dataset, in order to be comparable to my experiments in Chapter 9. I use the same parameters and environment from Tables 8.11 and 8.12, and I allow a maximum of 20 realization results (compared to 5 parsing results), again, to be consistent with my experiments in Chapter 9. Table 8.13 shows the parsing performance, generation performance, and rephrasing BLEU scores. For parsing performance, I report the coverage and the average results per item (RPI). For generation, I report the coverage relative to the items that parsed (since one cannot generate from items with no semantic representations), the absolute coverage, and the average results per item (cumulative over parse results). For BLEU, I report both the scores for both the first result and the oracle result, as described in Section 9.1. Note that the BLEU scores only consider those items that resulted in a rephrasing (i.e., ignoring those that failed to generate anything), and that capitalization differences are normalized prior to BLEU calculation.

The ERG performs best on the Tanaka Corpus in terms of coverage and BLEU scores, and worst on the Kyoto Corpus. The oracle BLEU for the Tanaka Corpus is in the 90s, meaning that it's very likely to get back the original string, given ideal semantic inputs.

8.5 The Bisem Corpus

Transfer rule extraction depends on the availability of semantic representations for both sides of a bitext. Since there is incomplete coverage for the Japanese and English sides of the bitext, the bilingual semantic (**bisem**) corpus will necessarily be smaller in size than the original bitext. Table 8.14 shows the number of successfully parsed items in both the

[13]The LKB (Copestake, 2002), however, has some limited ability to generate from unknowns.

	Parsing		Generation			BLEU	
Corpus	Cov	RPI	Rel Cov	Abs Cov	RPI	First	Oracle
Tanaka	96.4	4.5	95.9	92.4	50.9	69.1	91.7
Kyoto[a]	77.1	4.6	68.6	52.9	62.1	63.8	78.0
WordNet	92.7	3.9	73.8	68.4	47.2	56.3	81.5
Total[a]	84.3	4.4	75.0	63.2	55.0	62.7	81.1

[a] There were processing errors in 5 Kyoto-wiki subsets, preventing them from being used in the calculations, but the performance across subsets is fairly consistent, so the numbers should be accurate

Table 8.13: Generation performance on the development data

Japanese and English sides of the training data for each corpus, as well as the results per item (RPI), up to 2.[14]

	Items Parsed		RPI		Result Pairs	
Corpus	Jpn	Eng	Jpn	Eng	{1}×{1}	{1,2}×{1,2}
Tanaka	113,480	136,120	1.9	1.9	**109,769**	**416,069**
Kyoto	168,579	347,906	1.7	1.9	**148,227**	**483,675**
WordNet	118,267	155,656	1.8	1.9	**111,325**	**385,472**
Total	400,326	639,682	1.8	1.9	**369,321**	**1,285,216**

Table 8.14: Bisem counts

Of the ∼759k sentence pairs in the training data, about ∼400k of the Japanese side and ∼640k of the English side are usable. But in order for my system to be able to learn

[14] In fact I allowed up to 5 when parsing the training data, as with the other splits, but as I only consider up to 2 when training my models, I restrict my analysis here to 2 or fewer.

transfer rules from paired semantic representations, both sides must have a parse, which is the intersection of the two usable sets, which is ∼369k items.

Each source and target sentence can result in multiple semantic representations, so there can in fact be more semantic pairings than the intersection of successfully parsed items. This is not only useful for increasing the amount and variety of data, but also for helping to capture good subgraph mappings in the case that the first parse is not the best one. With up to 2 parse results per item, I can get up to 4 pairings for each item pair: ⟨1,1⟩, ⟨1,2⟩, ⟨2,1⟩, and ⟨2,2⟩. In Table 8.14, the {1}×{1} column shows the number of pairings if I take only the first result (i.e., the intersection of parsed items), and the {1,2}×{1,2} column shows how many pairings I get if I take up to 2 results on each side. I report both numbers because, while I only use the {1,2}×{1,2} data for training my system, the additional pairings still represent the same input sentences, and thus will not give me as much new information as a pairing from different sentences.

8.6 Chapter Summary

In this chapter I have described the three corpora I use to train and evaluate my system: the Tanaka Corpus, the Kyoto Wiki Corpus, and the Japanese WordNet Corpus. Each corpus exhibits different qualities such as sentence length, variety of vocabulary, duplicate counts, etc., depending on nature of its source. These differences are further expressed in the ability of the ERG and Jacy grammars to parse the sentences. By analyzing the generation performance of the English data, I can estimate the upper bound of performance from an ideal translation system using these grammars. Finally, as my methodology requires semantic representations for both sides of the bitext, the bisem corpus I can use for training contains 1,285,216 semantic pairings from 369,321 sentences.

Chapter 9
EXPERIMENTAL DESIGN

This chapter outlines the experiments used to evaluate my methodology. I first explain my use of evaluation metrics in Section 9.1, the pipeline processing parameters in Section 9.2, and how I prepare the data for the various baseline and experimental systems in Section 9.3. I describe two baseline experiments for comparison: the first (Section 9.4) uses the popular Moses system (Koehn et al., 2007) to do phrase-based SMT, and the second (Section 9.5) is a semantic-transfer system that builds on the JaEn grammar (Bond et al., 2011) with transfer rules automatically extracted by matching rule templates to bilingually-aligned n-grams of semantic predicates (Haugereid and Bond, 2011, 2012), hereafter referred to as the H&B system. Moses's near-ubiquity in statistical machine translation research makes this baseline system a good landmark for judging the performance of my system on a fixed corpus, while the H&B system extends the JaEn transfer grammar, like my systems, and is thus suitable for a direct comparison. I have two different experimental methods for extracting transfer rules: the first, called LPA (Section 9.6), uses bilingually-aligned n-grams of predicates, like H&B, but in contrast does not use templates (see Sections 6.1 and 6.2); and the second, called SGA (Section 9.7), builds a statistical model directly on bilingually-paired semantic subgraph.

9.1 Evaluation Metrics

Machine translation evaluation metrics compare the system output to a reference translation. BLEU (Papineni et al., 2001), the most commonly used metric, looks for n-gram matches between the strings. It is popular because it is intuitive, simple to use, and does not require resources beyond the system and reference strings, but it is also known for having several

deficiencies (Culy and Riehemann, 2003; Zhang et al., 2004; Ananthakrishnan et al., 2007). NIST (Doddington, 2002) is a variant of BLEU that gives more weight to "informative" (i.e., rare) n-grams, and has a different penalty formula for differences in string length. METEOR (Banerjee and Lavie, 2005; Lavie and Agarwal, 2007) aims to be more flexible than BLEU by giving more weight to unigram recall than precision and also by using stemming and inexact word matching (i.e., allowing synonyms rather than just exact matches). METEOR correlates with human judgment more closely than BLEU but the extra resources required inhibit it from scaling to lower-resource target languages. As I am translating to English, which has many such resources, METEOR is appropriate for my experiments, in addition to BLEU and NIST. I use the freely available `mteval-v14.pl` script[1] to calculate both BLEU and NIST, and I use version 1.5 of METEOR.[2]

All systems tested, baseline and my own, can produce multiple translations for their inputs. Moses uses an n-gram language model to rank its outputs, so the top-ranked output performs well against an n-gram metric like BLEU, and I therefore only record the first result from Moses. The JaEn-based systems (mine and the H&B baseline) do not use such a model[3] so instead I report two BLEU scores for each system, based on the two different translation selection methods described in Section 4.6. The *First* method selects the first translation for each input, while the *Oracle* method selects the translation with the highest BLEU score with respect to the reference translation. For the Oracle method, I use a smoothed variant of BLEU,[4] as it gives a useful score for individual sentences, unlike the standard BLEU which works best over larger samples. Note that Oracle selects using BLEU even when I evaluate using METEOR or NIST; if I used a different selection method for each metric, the numbers would not be comparable and the evaluation would be more confusing.

---

[1] `ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v14.pl`, also packaged with Moses 4.0. I additionally use the `wrap-mteval.pl` script available at `https://github.com/ufal/neuralmonkey`

[2] `https://github.com/cmu-mtlab/meteor`

[3] It is possible, but I do not explore the use of target language models in my experiments.

[4] Namely, NIST geometric sequence smoothing as implemented by the NLTK version 3.2.4, available at `https://github.com/nltk/nltk/releases/tag/3.2.4`.

Finally, I also ask human evaluators to compare the results of my system to a baseline for about 100 sentences. I give the annotators four pieces of information: the original Japanese sentence, the English reference translation, the output of a baseline system, and the output of one of my systems. The system outputs are randomized. The annotators have native or near-native English proficiency as well as reasonable proficiency in Japanese, which helps when the reference translation is not a close translation of the original sentence. The guidelines given to the annotators are as follows:

> For each item, the original Japanese and English reference sentences are given along with two (randomly ordered) translations. Choose the *most useful* translation from the two options, which is the one that most accurately conveys the original meaning. Minor grammatical mistakes or unnatural-sounding sentences are ok unless they severely impede understanding. If both equally translate the meaning, the one with the most fluent or natural English is preferred.
>
> The English reference may not always be reliable, so consult the Japanese sentence if you have doubts. Some translations may be unintentionally comical or inappropriate; a funny translation is not preferred over a boring one (unless they are otherwise equivalent). If both translations are equally good or bad in all respects, select the third option: "(no meaningful difference)". This should be used sparingly.
>
> For example, if the reference is "The dog chased the cat" and choice (a) is "The cat chased the dog" and (b) is "The dogs are chasing cat", (b) is preferred despite the differences in number, tense, and grammaticality. If (a) is "to be chasing?" and (b) is "hot dog", then option (c) "(no meaningful difference)" is ok, since neither option conveys the original meaning.
>
> There is a space at the bottom of the form for (optional) comments and questions.

Human evaluation is time-consuming, so it is only performed for one of my systems on the

test data.

9.2 Pipeline Parameters for Transfer-based Systems

The H&B baseline system and my experimental systems all use the same parsing, transfer, and generation software, with the only difference being the rules used in the transfer stage. I mitigate processing-environment sources of variation by fixing the parameters used by all systems to the same values so that the experiments are only testing the different approaches to transfer. Note that these parameters are not relevant for the Moses baseline system as the underlying software is different, but I use the same training, development, and test data, as well as the same data preprocessing steps where possible.

ACE[5] is used for processing, and Table 9.1 shows the parameters used for parsing, transfer, and generation.[6] The parameters are mostly the same for each process, but I use more generation results than parsing or transfer. In addition, parsing has a parameter for the maximum sentence length, which is irrelevant for the other tasks, and similarly generation has a parameter for disabling the MRS subsumption test. This MRS subsumption test is useful in the rephrasing task (generating from parsed rather than transferred MRSs) for ensuring that the MRS of a realized sentence is subsumed by the parsed MRS,[7] but for translation the test is often too restrictive, so I turn it off.

Using the same pipeline parameters for all systems helps to control for variation not directly due to the transfer rules. The parsing step, described as part of the data preparation in the following section, is done just once, and its results are reused across the different experiments.[8] Even though the generation parameters are fixed, generation has to be done

---

[5] `http://sweaglesw.org/linguistics/ace/`

[6] The parsing parameters are the same as in Table 8.11 in Section 8.3, but are repeated here for convenience.

[7] The details of why this might not be the case are irrelevant here.

[8] The performance of the parsing process can have per-run variability, as the processing speed, available memory, current load, etc. of the machine doing the parsing can cause the parser to hit a specified timeout or memory limit sooner or later, so by parsing once and reusing the results I can control for these sources of variation.

	Value		
Parameter	Parsing	Transfer	Generation
sentence length (tokens)	70	n/a	n/a
MRS subsumption test	n/a	n/a	no
results	5	5	20
timeout (seconds)	10	10	10
chart memory (MiB)	4096	4096	4096
unpacking memory (MiB)	4096	4096	4096

Table 9.1: ACE parameters for parsing, transfer, and generation

separately in each experiment because it operates on the output of transfer, which is the variable being manipulated.

9.3 Data Preparation

In order to keep the baseline and experimental systems comparable, I train and evaluate them on the same data (described in Chapter 8). In addition, I use mostly the same preprocessing steps for each system. While the Moses baseline uses the bitext view of the data, the JaEn-based systems (H&B, LPA, and SGA) use the **bisem** view, i.e., the subset of the data that has both source and target semantic representations produced by the ERG (Flickinger, 2000) and Jacy (Siegel et al., 2016) grammars.

Both H&B and LPA use a similar kind of predicate alignments but they involve slightly different filtering and preparation, so I run the aligner over each set separately. For each, I align the predicates by first creating a node-only (predicate) linearization of both sides of the bisem corpus (see Section 6.1.1), then by running Anymalign (Lardilleux et al., 2012) on the linearized training data. The bisem has up to 5 semantic representations for every sentence, which means there are up to 25 bilingual semantic pairs per sentence, however I

only use the first results of the source and target. In contrast, the Moses baseline cannot use the same alignments as it models lexical tokens and not predicates and SGA does its own alignments.

I let Anymalign run for 16 hours (following Haugereid and Bond 2012) and let it calculate the lexical weight in addition to translation probabilities.[9] The result is 7,766,440 alignments covering 1,183,584 unique source predicate phrases and 1,194,382 unique target predicate phrases.[10] With the prepared training data and alignments I can build the baseline and experimental systems.

9.4 Moses Baseline

The Moses statistical machine translation system (Koehn et al., 2007) is a large collection of software with many options and tunable parameters. The choice of translation model (whether phrase-based, hierarchical, or something else) is important, but performance is also significantly affected by pre- and post-processing of the data.

Shallow preprocessing techniques, such as recasing and tokenization, can have a large effect on performance as they can normalize the inputs to avoid data-sparsity issues. For languages with an upper and lower case dictinction in the orthography, a **truecase** model can be trained to predict what words, such as proper names, should be capitalized. This model can be used to normalize the training data. Tokenization also helps normalize the text, especially when there is extensive morphology (e.g., Japanese) or few encoded word delimiters like spaces (e.g., Chinese, or Japanese again). Even English, with its relatively low amount of inflectional morphology and use of spaces and punctuation between words, can benefit from tokenization.

For bilingual word alignment, Giza++ (Och and Ney, 2003) is frequently used with Moses, but other aligners, such as Anymalign (Lardilleux et al., 2012) and fast_align (Dyer

[9]The parameters I use for Anymalign are `--timeout=57600 --weight`.

[10]Anymalign works by continually selecting random subsets of the data until manually stopped or a threshold is reached, so subsequent runs with the same settings will produce different alignments.

et al., 2013), are also available. The word aligners themselves may require some parameters to be tuned to improve performance. The aligners produce a **phrase table**, which assigns forward and backward translation probabilities to bilingual n-gram pairs. The intersection of forward and backward alignments tends to have higher precision and lower recall, and the union of the two increases recall but lowers precision. The default method Moses uses, `grow-diag-final`, starts with the intersection of forward and backward word alignments, then adds additional points from the union of the forward and backward word alignments, and finally adds alignment points for previously unaligned source or target tokens. I use `grow-diag-final-and`, which is the same as `grow-diag-final`, except it only adds alignment points for unaligned tokens if both the source and target were previously unaligned. These alignments are then used to get a lexical translation table and extract phrase pairs.

A lexicalized reordering model is a separately-built model for discounting phrase translations during decoding based on the amount or kind of deviation from a monotonic (i.e., word-for-word) translation. Languages with very divergent word order, such as Japanese and English, can benefit from a more permissive reordering model. I use `msd-bidirectional-fe`, which is a word-based model that allows monotone, swap, and discontinuous (`msd`) orientations for reordering, considers both the previous and next phrases (`bidirectional`), and is conditioned on both the source and target languages (`fe`).

Finally, a target-side language model is trained so translation candidates can be weighted by their naturalness in the target language. The language model is monolingual, so it can be trained on potentially much more data than just the target side of the training bitexts.

In order to build a baseline system that's comparable to my own, I use the same datasets and tokenization for Japanese,[11] and set the maximum sentence length to 70, the same as for ACE `0.9.26`. I use Moses version 4.0 with the Experiment Management System (EMS).[12] The basic settings I use are listed in Table 9.2 and the full EMS configuration file is given in

[11]The English tokenization is done separately. The ERG uses its own REPP-based tokenizer (Dridan and Oepen, 2012), while Moses uses its own `tokenizer.perl`.

[12]See `http://www.statmt.org/moses/?n=FactoredTraining.EMS`

Parameter	Value
Maximum Sentence Length	70
Language Model	KenLM, order=5
Word Aligner	MGiza
Alignment Symmetrization	`grow-diag-final-and`
Lexicalized Reordering	`msd-bidirectional-fe`
Max Phrase Length	5
Tuning	MERT, nbest=100

Table 9.2: Basic settings for the moses baseline

Appendix A.1. These settings define a baseline Moses system that, while not fully optimized for the task, should provide competitive and comparable results.

9.5 Haugereid and Bond Baseline

Haugereid and Bond (2011, 2012) built upon the JaEn core transfer grammar by automatically extracting MRS transfer rule (MTR) instances from parallel corpora. The results of their extraction are included with the JaEn grammar by default,[13] but I reproduce their extraction process using my training data in order to make the resulting system, the H&B system, comparable to my experimental systems and to the Moses baseline.

Using the H&B templates (updated to current versions of Jacy and the ERG), the predicate alignments (described in Section 9.3), and the bisem training data, I extract both **single** rules, which have one predicate on the source side, and rules for **multi-word expressions** (MWEs), which have more than one predicate on the source side.[14] Without redoing the

[13] At `https://github.com/delph-in/jaen`

[14] Calling them multi-*word* expressions is perhaps a misnomer, because there is not necessarily a one-to-one correspondence between predicates and realized words; i.e., a single rule could match more than one word, and multiple predicates may match a single word.

extraction, this baseline would contain knowledge from sources that my systems could not make use of, thus obscuring the differences between my systems and the baseline. By rerunning the extraction process I also update rules that have become obsolete due to changes in the underlying English and Japanese grammars.

The H&B method initially extracts a large number of rules, many of which may turn out to be unhelpful, so the rules are filtered by their translation probability or their corpus frequency. Without filtering, the transfer grammar may be too large to compile, as discussed in Section 7.2, but even if it compiles the performance could suffer greatly. The extra rules increase the search space for transfer, which can make timeouts more likely. Dispreferred transfers can push the preferred ones past the result cutoff, which is further complicated by the fact that ACE `0.9.26` does not do transfer reranking. In H&B, the single (i.e., one-to-N) transfer rules generally must have a forward translation probability of 0.1 and the MWE rules must have 0.01, but there are also hard-coded thresholds for specific rule templates. Haugereid and Bond decided these probability thresholds by manual inspection of the extracted rules. In addition to the probability threshold, rules with a corpus frequency of 1 are culled.[15]

After filtering, I extract 38,696 single rules and 15,018 MWE rules. The distribution of these rules according to the number of source and target predicates that are specified in the MTRs[16] is shown in Table 9.3. Cells with a number (even 0) represent those for which a template is defined. Single rules are by far the most numerous, and in general there are more rules for the lower predicate counts. The higher the predicate count, the more specific a hand-written template will be, and therefore fewer instances will match such templates.

---

[15]Haugereid and Bond (2012) uses both Giza++ and Anymalign alignments. In Anymalign, which I use for H&B, the *frequency* is not the absolute corpus frequency, but the number of times it appears in randomly selected subsets during the training procedure, so it may differ across runs.

[16]Unlike the counts for my systems in Tables 9.5 and 9.10, which are distributed by the number of predicates the resulting transfer rules will match or insert, the H&B predicate counts do not consider predicates defined in the transfer rule types; thus, the actual number of predicates affected may be higher than the numbers used in Table 9.3. The difference is due to a limitation in the way the rules are encoded for H&B.

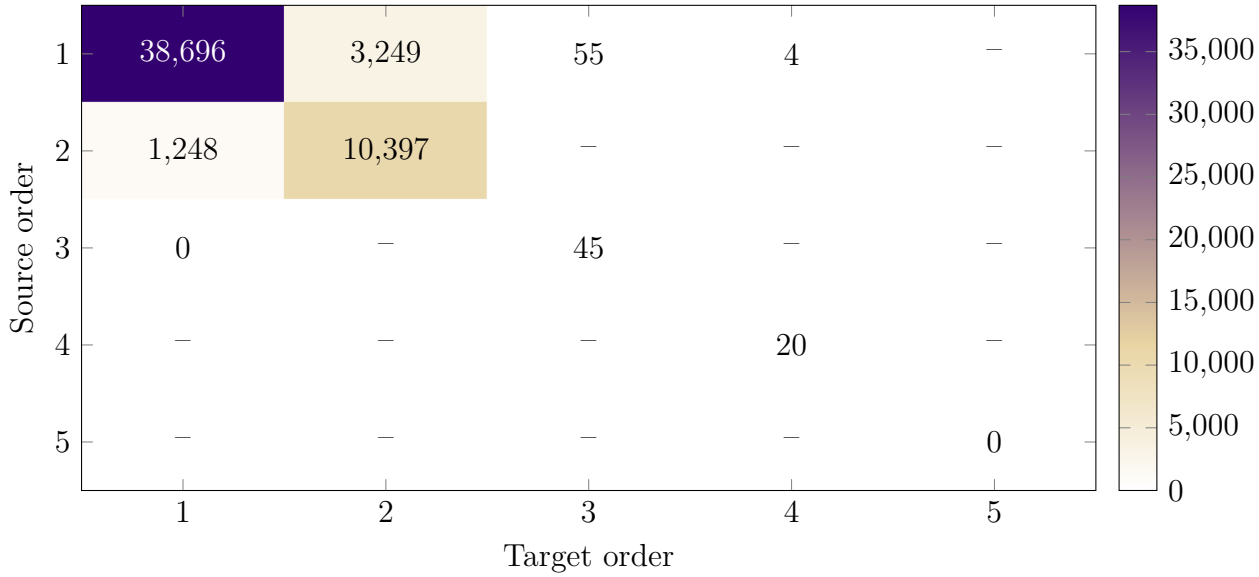


Table 9.3: Extracted H&B rule counts by specified predicate counts

Haugereid and Bond (2011, 2012) defined a set of templates for matching semantic patterns and I minimally update and then use those templates for re-extracting transfer rules from my datasets. The result is 38,696 single rules and 15,018 MWE rules, which I use to build the H&B transfer grammar.

9.6 LPA: Bilingually-aligned Predicate Phrases

This first experimental system, called LPA, is an extension of Haugereid and Bond 2012 that does not use rule templates but instead uses properties of the semantic graphs, as described in Section 6.1. I use the alignments described in Section 9.3 and the bisem training data for rule extraction.

9.6.1 Linearization and Extraction Parameters

When linearizing the predicates for alignment, I drop some quantifiers (as explained in Section 5.7.1) as well as any other nodes with predicates listed in Table 9.4, as they are generally inserted by the system and can be reinserted by the hand-written rules. During

projection and extraction, I require the top nodes in the source and target subgraphs to have the same variable type, and I limit the source:target subgraph order ratio to not exceed 1:3 or 3:1.[17]

Predicate	Jacy	ERG
`udef_q`	✓	✓
`pronoun_q`		✓
`number_q`		✓
`proper_q`		✓
`def_q`	✓	✓
`_wa_d`	✓	
`parg_d`		✓
`pron`	✓	✓
`compound`	✓	✓
`unknown`		✓
`unknown_v`	✓	
`unknown_v_cop`	✓	
`nominalization`	✓	✓

Table 9.4: Dropped predicates for linearization; checkmarks indicate if the predicate is relevant for a grammar

The `compound` and `nominalization` predicates, which have the same form and function in both the ERG and Jacy, are in fact useful to include in subgraph pairs but only if their arguments are fully satisfied. If they are included in the linearization, the aligner may find predicate phrase pairs that include `compound` or `nominalization`, but not all of their arguments, so in response I treat them specially. They are dropped during linearization, but

[17]The order ratio prevents 1×4, 1×5, 1×6, 4×1, 5×1, and 6×1 orders.

may be automatically inserted during extraction if all of their arguments are present in a predicate phrase pair, as described in Section 6.1.2.

9.6.2 Subgraph Extraction and Filtering

The initial extraction obtains 749,089 paired subgraphs. I filter out those that have a source or target graph order greater than 6, an Anymalign frequency of 1, or a forward translation probability of less than 0.1. Note that unlike H&B I do not use different probability thresholds for different graph sizes or patterns. I also remove pairs where the target side includes unknowns, i.e., predicates not included in the target grammar's lexicon, as they would not lead to any realizations (as discussed in Section 8.4). After these filtering steps, 359,407 pairs remain.

Table 9.5 shows how many unique rules were extracted from the training data. The rows correspond to the order of the source subgraphs and the columns correspond to the order of the target subgraphs. As with H&B, there are many more single rules than any other order pairing and the number of single rules is similar to that of H&B. LPA, however, extracts many more MWE rules of various sizes, with a clear trend along the diagonal. The large number of extracted rules is encouraging for improving transfer coverage, but the challenge is to make sure that only high-quality rules are used so that the bad ones do not hide the good results and that the enlarged search space does not lead to increased timeouts. I apply the strategy described in Section 7.2.2 for filtering the rules.

Table 9.6 shows the counts of the isomorphic rules only.[18] As the isomorphism constraint (see Section 5.3.2) requires the graphs on both sides to have the same order, cells in Table 9.6 correspond to the diagonal of Table 9.5, e.g., for 2×2 order 16,628 of 26,353 rules were isomorphic.

[18]The only single (one-to-one) rule that was not isomorphic mapped a regular nominal predicate to one with a constant argument; constant arguments are node properties (thus it does not introduce a second node) but their presence is considered for isomorphism comparisons.

Table 9.5: Extracted LPA rule counts by subgraph order

9.6.3 System Configurations

I select subsets of the bilingual subgraph pairs in order to create experiments for LPA. Consider X to be the transfer rule store, i.e., the set of all bilingual subgraph pairs $\langle g_s, g_t \rangle$ extracted by predicate phrase projection. I define Eq. (9.1) to select pairs where the subgraphs have a specified source order between h and i and target order between j and k.[19] Eq. (9.1) is in set-builder notation and is interpreted as: the set of all pairs $\langle g_s, g_t \rangle$ in X such that the size of the set of vertices in graph g_s is greater than or equal to h and less than or equal to i, and the size of the set of vertices in graph g_t is greater than or equal to j and less than or equal to k. I then define Eqs. (9.2) to (9.4) as variations of Eq. (9.1) that allow a source or target range to be a single number meaning a graph order of exactly that number. For example, $X_{1,1}$ is the set of one-to-one pairs, $X_{2,1-2}$ is the set of pairs that have

[19] V is a function that returns the set of nodes, or vertices, thus $|V(g)|$ is the order of the graph g.

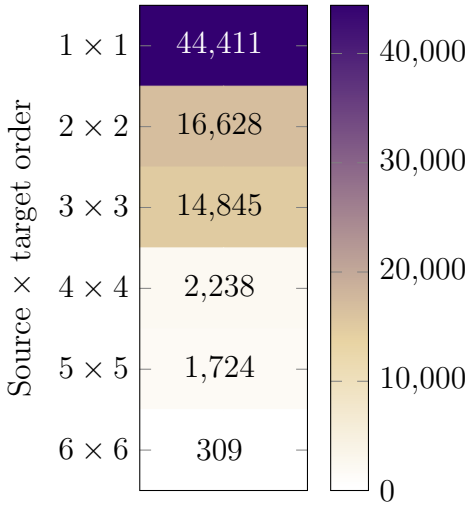


Table 9.6: Extracted LPA isomorphic rule counts by subgraph order

2 predicates on the source side and between 1 and 2 predicates on the target sidel, etc.

$$X_{h\text{–}i,j\text{–}k} = \{\langle g_s, g_t\rangle \in X : (h \leq |V(g_s)| \leq i) \text{ and } (j \leq |V(g_t)| \leq k)\} \tag{9.1}$$

$$X_{i,j\text{–}k} = X_{i\text{–}i,j\text{–}k} \tag{9.2}$$

$$X_{h\text{–}i,j} = X_{h\text{–}i,j\text{–}j} \tag{9.3}$$

$$X_{i,j} = X_{i\text{–}i,j\text{–}j} \tag{9.4}$$

Using these selectors, I define four sets of experiment configurations (S, M, P, and O) for testing the performance of my rule extraction and filtering. The first two sets (S and M) are augmented with partial H&B rules in order to separately evaluate the contribution of my single and MWE rules, respectively, from the H&B baseline. Of these two sets, the first consists of three experiments, S1–S3, which use my single rules, i.e., a source graph size of 1 and a target graph size from 1 to 3, along with H&B's MWE rules. The second set consists of five experiments, M2–M6, one for each maximum MWE graph size from 2 to 6, using my

	Extracted Rule Sets		Extracted Rule Counts	
Name	Single	MWE	All	TC-dev
LPA-S1	$X_{1,1}$	H&B	72,937	21,638[a]
LPA-S2	$X_{1,1–2}$	H&B	84,239	23,885[a]
LPA-S3	$X_{1,1–3}$	H&B	96,605	24,709[a]
LPA-M2	H&B	$X_{2,1–2}$	98,160	64,521[b]
LPA-M3	H&B	$X_{2–3,1–3}$	210,659	106,591[b]
LPA-M4	H&B	$X_{2–4,1–4}$	298,878	148,125[b]
LPA-M5	H&B	$X_{2–5,1–5}$	394,060	186,732[b]
LPA-M6	H&B	$X_{2–6,1–6}$	470,761	219,377[b]
LPA-P2	$X_{1,1–2}$	$X_{2,1–2}$	128,685	34,692
LPA-P3	$X_{1,1–3}$	$X_{2–3,1–3}$	253,550	77,586
LPA-P4	$X_{1,1–4}$	$X_{2–4,1–4}$	341,772	119,120
LPA-P5	$X_{1,1–5}$	$X_{2–5,1–5}$	436,954	157,727
LPA-P6	$X_{1,1–6}$	$X_{2–6,1–6}$	513,655	190,372
LPA-O2	$X_{1,1}$	$\textsc{Iso}(X_{2,2})$	82,778	18,776
LPA-O3	$X_{1,1}$	$\textsc{Iso}(X_{2–3,2–3})$	101,588	22,771
LPA-O4	$X_{1,1}$	$\textsc{Iso}(X_{2–4,2–4})$	104,157	23,754
LPA-O5	$X_{1,1}$	$\textsc{Iso}(X_{2–5,2–5})$	106,121	24,125
LPA-O6	$X_{1,1}$	$\textsc{Iso}(X_{2–6,2–6})$	106,449	24,251

[a] Only single rules were refined; H&B MWE rules remained constant.
[b] Only MWE rules were refined; H&B single rules remained constant.

Table 9.7: Experimental configurations for LPA

MWE rule and H&B's single rules. The second two sets of experiments use my LPA rules only; P2–P6 use all available rules, and O2–O6 restrict the MWE rules to isomorphic MRS subgraphs. The O set uses the ISO() function, which performs a structural isomorphism comparsion as defined in Section 5.3.2, on the selected rules so that only isomorphic pairs are included. These configurations are listed in Table 9.7 along with the total number of rule counts and the number of rules relevant to the development data of the Tanaka Corpus (Tanaka, 2001).[20]

9.6.4 Summary

LPA is a method of transfer rule extraction that is similar to the H&B method but operates without rule templates. Not being constrained to a limited number of hand-written templates allows it to extract many more MWE rules than H&B, however it likely has more noise as the rule filtering is less strict (considering the templates as a kind of filter). The quantity of extracted rules enables me to experiment with different system configurations, which I do on the basis of source and target graph order and isomorphism of the paired graphs.

9.7 SGA: High-frequency Coincident Subgraphs

The second experimental system does not rely on linearized predicates but instead aligns extracted subgraphs directly, as described in Section 6.2. I use the bisem training data described in Section 9.3 for subgraph extraction. This method initially yields a far greater number of subgraph pairs than LPA, so I must apply more aggressive filtering in order to remove the noise and extract a reasonable number of aligned subgraph pairs. For example, (22) is an example sentence (and its translation) from the Tanaka Corpus and Figs. 9.1 and 9.2 show some of the enumerated subgraph pairs where the single node for 海王星 *Kaiousei* "Neptune" is the source subgraph. Fig. 9.1 is the correct subgraph pairing that is

[20]Note that the S1–S3 and M2–M6 sets use H&B MTR rules which, being in the MTR format, are not compatible with the method I have for selecting corpus-relevant subgraph pairs (see Section 7.2.1), so the values of the TC-dev column for those sets are not comparable with those for P2–P6 and O2–O6.

enumerated by SGA but there are many incorrect pairings that would also be enumerated, as shown by the partial list in Fig. 9.2.

(22) 海王星 は 太陽系 の 8 番目 の 惑星 だ
Kaiousei -ha taiyoukei -no hachi -banme -no wakusei da
Neptune -TOP solar.system -GEN 8 -ORD -ADN planet COP
"Neptune is the eighth planet of the solar system." [jpn]

`(x0 / named` `  :carg "kaiousei_1")`	`(x0 / named` `  :carg "Neptune")`

Figure 9.1: Correct subgraph pairing for 海王星 *Kaiousei* "Neptune"

`(x0 / named` `  :carg "kaiousei_1")`	`(x0 / _planet_n_1)`

`(x0 / named`
`  :carg "kaiousei_1")`
	`(x0 / _planet_n_1`
	`  :ARG1-EQ-of (e1 / ord`
	`  :carg "8")`
	`  :ARG1-EQ-of (e2 / _of_p)`
	`  :RSTR-H-of (u4 / _the_q))`

`(x0 / named`
`  :carg "kaiousei_1")`
	`(x0 / _system_n_of`
	`  :ARG1-EQ-of (e1 / _solar_a_1)`
	`  :RSTR-H-of (u2 / _the_q))`

Figure 9.2: Incorrect subgraph pairings for 海王星 *Kaiousei* "Neptune"

In this section I will explain how I use prefiltering to exclude subgraph pairs based on graphical patterns or properties, a weighted-ϕ^2 filter (discussed in Section 6.2.4) to exclude pairs that are not likely to be translationally equivalent (such as those in Fig. 9.2), and a

symmetric translation probability filter to further reduce the size of the model and to inform a partial ordering of MTRs in the resulting transfer grammars.

9.7.1 Enumeration Parameters

This method requires filtering at several stages. In the subgraph enumeration stage, I set the depth limit to 3 (where a depth of 0 is a single node) and drop nodes with predicates matching those in Table 9.8. When pairing the enumerated subgraphs, I require that the type of the top nodes is the same. I prefilter any pairs with a source or target subgraph matching a corresponding prefilter pattern as defined in Table 9.9. The prefilter patterns were chosen by looking at the most common subgraphs initially extracted and selecting those that did not appear useful.

Predicate	Jacy	ERG
`udef_q`	✓	✓
`pronoun_q`		✓
`number_q`		✓
`proper_q`		✓
`def_q`	✓	✓
`_wa_d`	✓	
`parg_d`		✓

Table 9.8: Dropped predicates for enumeration; checkmarks indicate if the predicate is relevant for a grammar

After building the statistical model, I keep single rules (those with a source order of 1) with a weighted-ϕ^2 value of 0.01, and MWE rules with a weighted-ϕ^2 value of 0.1. I then calculate the forward and backward translation probabilities on the remaining subgraph pairs and, for each group of pairs sharing the same source subgraph, I keep the top

Jacy	ERG	Notes
(e0 / unknown_v		With or without relations
	(* / _and_c)	Any variable type
	(* / _or_c)	
	(* / _but_c)	
	(* / implicit_conj)	
(x0 / pron)	(x0 / pron)	Don't pair solitary pronouns
(e* / compound)	(e* / compound)	Missing at least one argument
(x* / nominalization)	(x* / nominalization)	
(e* / _no_p)	(e* / _of_p)	
(e* / _ni_p)	(e* / _in_p)	
(e* / _de_p)	(e* / poss)	
(e* / coord_c)	(e* / _for_p)	
(e* / _to_p_with)	(e* / appos)	
(e* / _to_p_and)	(e* / _to_p)	
(e* / _ya_c)	(e* / _with_p)	
(e* / _toshite_p)	(e* / _on_p)	
	(e* / _as_p)	
	(e* / _from_p)	
	(e* / _at_p)	
	(e* / _after_p)	

Table 9.9: Subgraph prefilters

5 based on the symmetric translation probability, after filtering out pairs where the target has unknown nodes. Note that the weighted-ϕ^2 filter has an absolute threshold, designed to exclude pairings that likely are not translationally equivalent. In contrast, the symmetric translation probability filter's purpose is the reduce the size of the model instead of deciding what is translationally equivalent, so it uses an relative threshold: it keeps the top 5 per group regardless of what the absolute value of the probability is. The symmetric translation probability is also used to provide a partial ordering of rules in the transfer grammar, along with other criteria, such as graph order (see Section 7.2.2).

9.7.2 Subgraph Enumeration, Pairing, and Filtering

The intial enumeration, pairing, and prefiltering of SGA subgraphs yields 28,387,278 subgraph pairs. This number is all observed pairs and not just those that are likely translationally equivalent, so I expect the number to drop substantially when I apply the secondary filters. The subgraph enumeration method (described in Section 6.2) limits the depth of the subgraphs, but not their order. The highest source and target orders are 16 and 18, respectively, but those with a source or target order over 6 are relatively few—just 608,222 of the 28,387,278 (2.1%)—leaving 27,779,056 subgraphs with source and target orders between 1 and 6.

Table 9.10 shows the distribution of unique subgraph pairs for source and target orders between 1 and 6. As with LPA, the 1×1 cell has the most pairs. Unlike LPA, however, the trend does not follow the diagonal, but is instead radial from the 1×1 cell. This stands to reason, as single nodes are repeated across a corpus more often than any larger subgraphs. One cell does not follow this pattern, where the 1×3 (source order is 1 and the target is 3) has more pairs than the 1×2, and this is likely due to the prefiltering, e.g., because many of the predicates in the prefilter patterns have binary arity, so subgraphs including them that would not be filtered have an order of 3 or greater. Also note that there is a strong tendency for subgraphs that have order 3, which is likely due to there being many binary predications (e.g., transitive verbs, compounds, coordination, prepositions, etc.). See Section 11.3 for

some discussion of the distribution of different subgraph topologies.

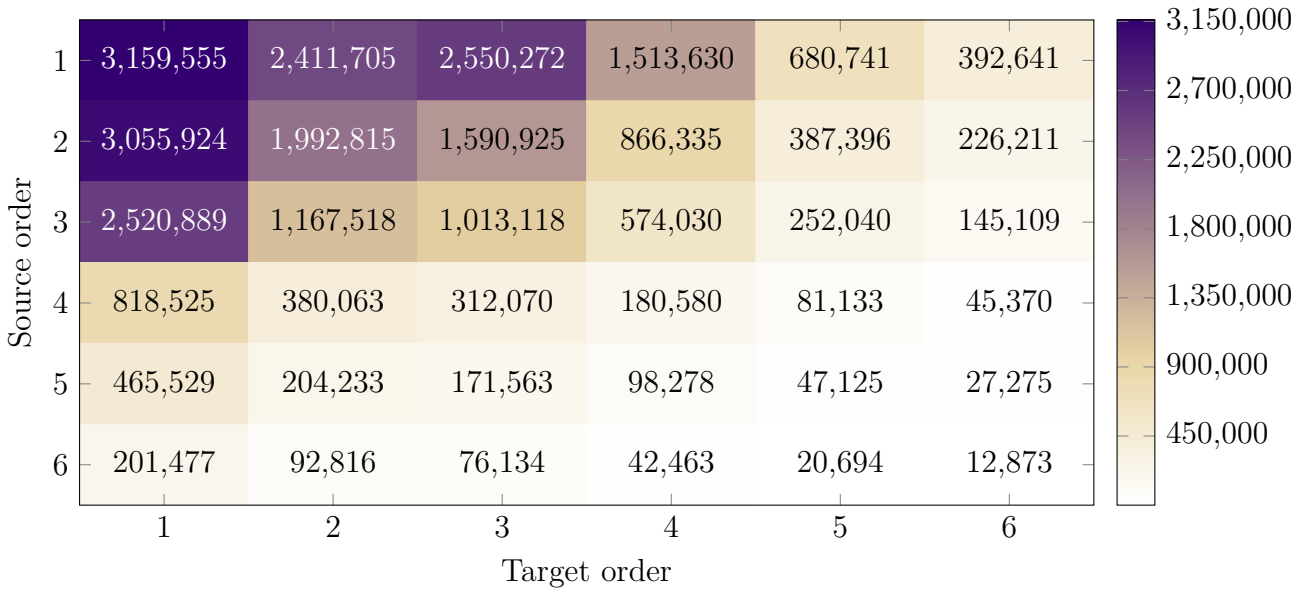
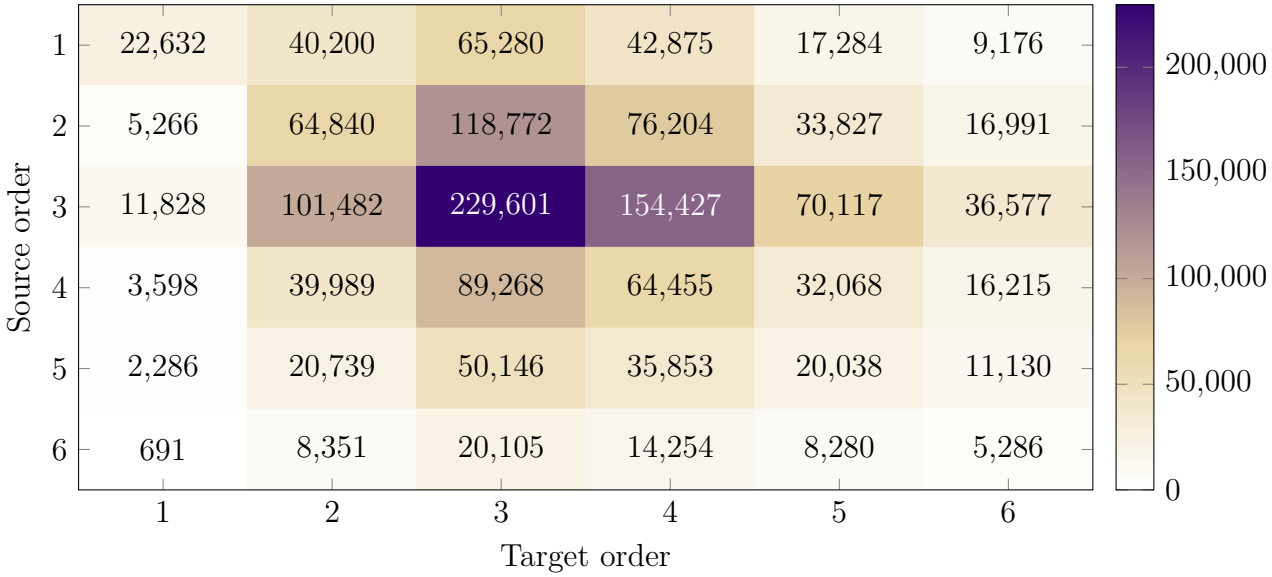


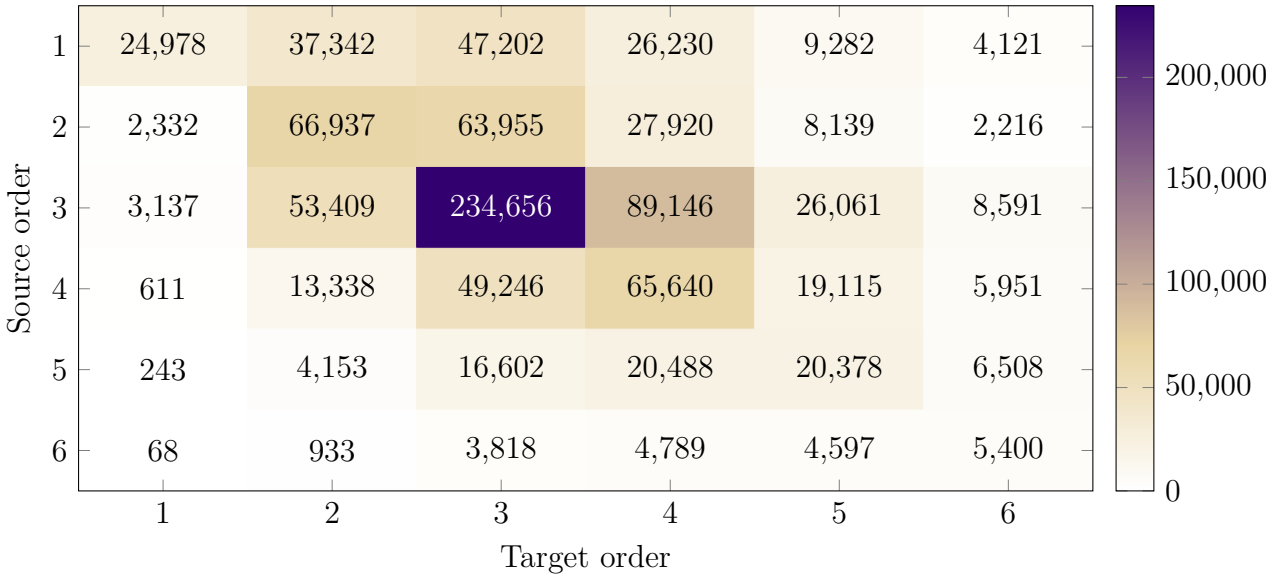
Table 9.10: SGA subgraph pair counts by subgraph order

As discussed in Section 6.2.4, I use a weighted ϕ^2 filter for ranking subgraph pairs. In order to motivate that choice, I first show pairs filtered by the unweighted ϕ^2 score in Table 9.11a. The numbers represented in that table were filtered keeping single rules with a ϕ^2 threshold of 0.01 and MWE rules with ϕ^2 threshold of 0.1, after which I calculated forward and backward translation probabilities, grouped rules by the source subgraph, and kept the top 5 per group based on their symmetric translation probabilities. The total number of remaining pairs is 1,560,131; a reduction of 94.4%. Note that the number of 1×1 rules has been greatly reduced relative to other cells, and the radial dispersion now centers around the 3×3 cell.

Table 9.11b shows the distribution of subgraph pairs filtered as before but using the weighted ϕ^2 ranking. Translation probabilities were calculated after the weighted ϕ^2 filtering and used for the symmetric translation probabilty filter. The number of remaining pairs is 977,532; a reduction of 96.5% from the original amount. The weighting skews the distribution



(a) SGA subgraph pair counts, filtered by ϕ^2



(b) SGA subgraph pair counts, filtered by weighted-ϕ^2

to follow the diagonal more closely, which is what I want, as it is easier to do bilingual variable binding (see Section 7.3.2) when the subgraphs are closer in size and shape. Note that despite the overall reduction of subgraph pairs, some cells have increased their counts compared to the normal ϕ^2 filtering. This is because the filtering based on the symmetric translation probability happens after the weighted ϕ^2 filtering, which changes the probabilistic model, thus resulting in more source subgraphs that have at least 5 pairs. The counts of isomorphic subgraph pairs is given in Table 9.12.

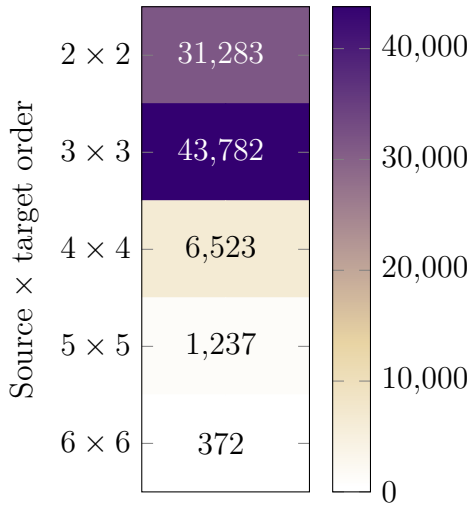


Table 9.12: Extracted SGA isomorphic subgraph pair counts

9.7.3 System Configurations

Just as with LPA, I select subsets of the extracted subgraph pairs in order to create experiments. The experimental configurations I create for SGA follow the same pattern as LPA, and I use the same selectors (Eqs. (9.1) to (9.4)) for dividing up the pairs by graph order. The difference is that I'm using the pairs extracted for SGA, which, unlike LPA, has pairs for the off-diagonal corners, thus allowing me to define a full 6 configurations of the S-series. Table 9.13 lists the configurations used for SGA with the total number of extracted pairs and the subset that is relevant for the Tanaka Corpus development set.

	Extracted Rule Sets		Extracted Rule Counts	
Name	Single	MWE	All	TC-dev
SGA-S1	$X_{1,1}$	H&B	36,402	17,280[a]
SGA-S2	$X_{1,1–2}$	H&B	76,602	21,209[a]
SGA-S3	$X_{1,1–3}$	H&B	141,882	26,053[a]
SGA-S4	$X_{1,1–4}$	H&B	184,757	29,448[a]
SGA-S5	$X_{1,1–5}$	H&B	202,041	30,609[a]
SGA-S6	$X_{1,1–6}$	H&B	211,217	31,274[a]
SGA-M2	H&B	$X_{2,1–2}$	110,050	60,589[b]
SGA-M3	H&B	$X_{2–3,1–3}$	571,733	214,317[b]
SGA-M4	H&B	$X_{2–4,1–4}$	999,674	353,683[b]
SGA-M5	H&B	$X_{2–5,1–5}$	1,264,748	435,613[b]
SGA-M6	H&B	$X_{2–6,1–6}$	1,402,628	476,443[b]
SGA-P2	$X_{1,1–2}$	$X_{2,1–2}$	132,938	28,084
SGA-P3	$X_{1,1–3}$	$X_{2–3,1–3}$	659,901	186,656
SGA-P4	$X_{1,1–4}$	$X_{2–4,1–4}$	1,130,717	329,417
SGA-P5	$X_{1,1–5}$	$X_{2–5,1–5}$	1,413,075	412,508
SGA-P6	$X_{1,1–6}$	$X_{2–6,1–6}$	1,560,131	454,003
SGA-O2	$X_{1,1}$	iso($X_{2,2}$)	53,091	12,096
SGA-O3	$X_{1,1}$	iso($X_{2–3,2–3}$)	96,315	23,584
SGA-O4	$X_{1,1}$	iso($X_{2–4,2–4}$)	102,796	24,849
SGA-O5	$X_{1,1}$	iso($X_{2–5,2–5}$)	104,030	25,144
SGA-O6	$X_{1,1}$	iso($X_{2–6,2–6}$)	104,401	25,169

[a] Only single rules were refined; H&B MWE rules remained constant.
[b] Only MWE rules were refined; H&B single rules remained constant.

Table 9.13: Experimental configurations for SGA

9.7.4 Summary

SGA builds its own statistical model over enumerated subgraphs, so it initially considers many times more subgraph pairs than LPA. Through multiple stages of aggressive filtering (Sections 9.7.1 and 9.7.2) I get the number of pairs down roughly to the same level as with LPA. For ease of comparison, I create a series of experimental configurations similar to that of LPA but for SGA I am also able to do the full set of S configurations.

9.8 Chapter Summary

I have described two methods (Sections 9.6 and 9.7) for augmenting transfer grammars for machine translation, their parameters for construction (Sections 9.6.1 and 9.7.1), and a series of experimental configurations for evaluating their performance (Sections 9.6.3 and 9.7.3). The Moses baseline (Section 9.4) shows how a popular alternative translation paradigm performs, while the H&B baseline (Section 9.5) exists as the prior state of the art in the same transfer-based paradigm. The experimental configurations are designed to show, by their relative peformance, if my extracted MTRs are increasing transfer coverage or improving evaluation scores compared to those of the H&B baseline system. The Oracle evaluation method (Section 9.1) will show the upper bound of my systems' performance on the BLEU metric, which is the primary means of comparing to the Moses baseline. Once a high-performing system is identified, I ask human evaluators to select the best translations from either my system or the Moses baseline. In Chapter 10, I show the results of my experiments and in Chapter 11 I analyze those results.

Chapter 10

RESULTS

In Chapter 9 I described the baseline and experimental systems I use to evaluate my methodolgy and in this chapter I report the results of those experiments. For the transfer-based systems I report the transfer and generation coverage.[1] For all systems I report the BLEU scores over all outputs. For the transfer-based systems I report separate BLEU scores for the First and Oracle selection methods, as discussed in Section 4.6.[2] The outputs of each transfer-based system might cover the same input items as the other systems, so these BLEU scores are not comparable to each other and are only to be used as a rough estimate of an individual system's or configuration's quality. I therefore take the **intersective subset** of each experimental system (i.e., all LPA configurations plus the baselines, then all SGA configurations plus the baselines), for which the evaluation metrics are comparable. I compute the BLEU, NIST, and METEOR metrics over these subsets, separating the First and Oracle selections for transfer-based systems, as before. To help avoid confusion, I call the BLEU scores reported over all outputs **All-BLEU** (in Section 10.1 I also similarly report All-NIST and All-METEOR on the baseline systems, since I do not compare the baselines to each other with intersective evaluation), while for the intersective subsets I just call the metrics by their original names. Note that all systems, including Moses and H&B, will report slightly different numbers for the different subsets described above (i.e., the full and intersective subsets for both LPA and SGA).

In Section 10.1 I show the coverage for H&B and evaluation scores for both H&B and

[1]Transfer may yield partially trasferred MRSs for most items but I ignore those items when calculating coverage as they cannot be used in generation.

[2]While Moses has the option to return the top-n translations, I do not explore this space in my evaluation, and I only report the top-1 results.

Moses over the development data. Section 10.2 has the coverage and evaluation scores for LPA, and Section 10.3 has the same for SGA, both over development data. In Section 10.4 I select the best configuration from LPA and SGA for comparison[3] and evaluate using the test data. Examples of translations from the baselines and from my experimental systems over the development data are given in Section 10.5. Discussion of the results and error analysis is covered in Chapter 11.

10.1 Baseline Development Results

The Moses baseline system does not use the transfer pipeline and it is able to provide a translation for every sentence given to it, so there are no coverage numbers to report. The H&B system, however, does have imperfect coverage, so I report the pipeline transfer and generation coverage in Table 10.1. A component's relative coverage in Table 10.1 is the proportion of successfully processed items from the inputs to the component, while the absolute coverage is the proportion of successfully processed items from all pipeline inputs. For example, there are 4,500 total items in the Tanaka development set, and 3,572 (79.4%) of the source side items have at least one parse, so 79.4% is the upper bound of the absolute coverage. Of the parsed items, 995 have at least one transfer, which is 27.9% of 3,572 and 22.1% of 4,500. The numbers for generation are computed similarly, with 995 as the denominator for the relative coverage. Since generation is the last step in the pipeline that can lose items,[4] its absolute coverage is also the end-to-end translation coverage. This means that only 13.0% of the 4,500 items, or 585 items, resulted in a translation.

Table 10.2 shows the BLEU, NIST, and METEOR scores over all 4,500 results for the Moses baseline system. The H&B results, using both First and Oracle selection for all results, are shown in Table 10.3. The BLEU score reported for Moses, 21.81, is reasonable for the relatively small amount of training data for an SMT system but this score will change as it is

[3]The best configurations for comparison are not necessarily the ones with the highest scores; see Section 10.4 for more details.

[4]The last step, selection, just picks one of the available realizations in a hypothesis set.

Transfer		Generation	
Relative	Absolute	Relative	Absolute
27.9	22.1	58.9	13.0

Table 10.1: H&B transfer and generation coverage for the development set

computed over different subsets of the data in comparison to the output of my systems. For H&B, note that there is a very large difference between the First and Oracle in BLEU scores but a relatively smaller difference for METEOR, which shows the sensitivity of BLEU to word order, synonyms, and morphological variation. The Oracle BLEU score of 24.09 should not be compared with the BLEU of Moses, as they are for different subsets of the data and the items for which H&B had coverage may contain simpler or cleaner data. Comparable scores will be given when showing LPA and SGA results in the following sections.

BLEU	NIST	METEOR
21.81	6.04	27.42

Table 10.2: Moses evaluation results over the development data

All-BLEU		All-NIST		All-METEOR	
First	Oracle	First	Oracle	First	Oracle
10.52	24.09	3.85	5.45	26.52	31.47

Table 10.3: H&B evaluation results over the development data

10.2 LPA Development Results

There are 18 configurations for LPA. Tables with numerical values are useful for fine-grained comparison but they are not ideal for viewing the trends across all configurations. Therefore I present in Fig. 10.1 a visualization of the coverage and BLEU scores for all configurations and tables of results for the top configurations in Tables 10.4 and 10.5. The top configurations are the ones from each set with the highest intersective Oracle BLEU scores.

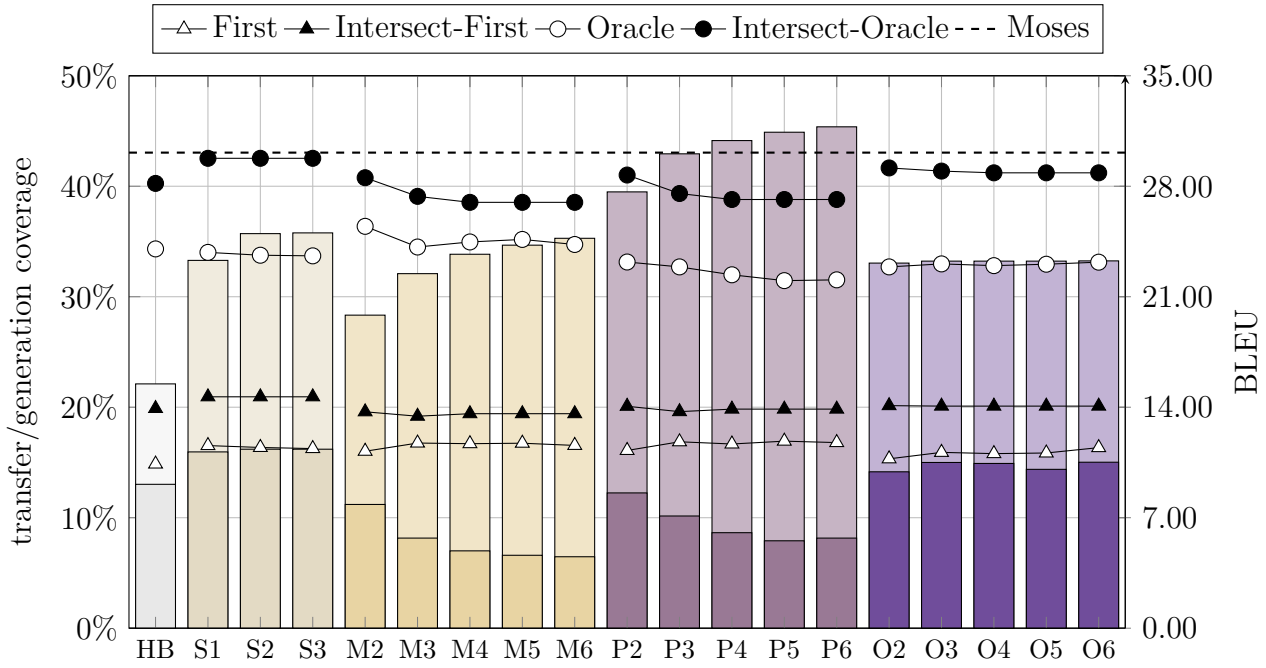


Figure 10.1: Coverage, All-BLEU, and Intersective-BLEU results for all LPA configurations

In Fig. 10.1 the different configuration set are grouped together so the trends for increasing graph order can be more easily seen. For each configuration there are two overlapping bars. The taller, lighter-colored bars are the absolute transfer coverage, while the shorter, darker bars are absolute generation coverage. The proportion of the lighter bars occupied by the darker bars is the relative generation coverage.[5] Note that the y-axis for coverage is given

[5]The relative transfer coverage is not shown, but the parsing coverage (79.4%) is constant for all systems, so the differences between configurations would be the same.

on the left side, ranging from 0% to 50%.[6] Overlaid on top of the coverage chart is a line plot of the BLEU scores, using triangle markers for First and circles for Oracle selection. Filled-in triangles and circles are used for the scores on the intersection of all configurations. The y-axis for BLEU scores is on the right side, ranging from 0.00 to 30.00.

	% Coverage					
	Transfer		Generation		All-BLEU	
System	Rel	Abs	Rel	Abs	First	Oracle
H&B	27.9	22.1	**58.9**	13.0	10.52	24.09
LPA-S2	45.0	35.7	**45.4**	**16.2**	11.45	23.81
LPA-M2	35.7	28.3	39.5	11.2	11.21	25.46
LPA-P2	**49.8**	**39.5**	31.0	12.2	11.26	23.20
LPA-O2	41.6	33.0	42.8	14.2	11.14	23.08

Table 10.4: Coverage and All-BLEU for top LPA configurations

The coverage and All-BLEU scores are given in Table 10.4 along with the H&B values, for comparison, copied from Section 10.1. The evaluation metrics—BLEU, NIST, and METEOR—on the intersection of all configurations is given in Table 10.5. For both tables, I show in bold the value of the best configuration of my systems. If a baseline has a better value, I also show in bold the best performing baseline (H&B or Moses). I do not bold the highest numbers in the All-BLEU columns as the numbers are not comparable; instead see the intersective BLEU scores in the Table 10.5.

[6]Reducing the upper bound to 50% makes the differences more pronounced.

	BLEU		NIST		METEOR	
System	First	Oracle	First	Oracle	First	Oracle
Moses	**30.13**		**5.21**		**31.16**	
H&B	13.92	28.18	3.56	5.03	26.70	32.09
LPA-S2	**14.66**	**29.77**	**3.68**	**5.07**	27.33	**32.54**
LPA-M2	13.71	28.54	3.59	5.00	26.97	31.88
LPA-P2	14.06	28.71	3.66	4.97	27.29	32.22
LPA-O2	14.10	29.16	3.67	5.02	**27.38**	32.45

Table 10.5: Evaluation results for top LPA configurations over the intersection of development translations

10.3 SGA Development Results

There are 21 configurations for SGA, so, similar to LPA, I present the results, along with the H&B results, in a bar chart for coverage overlaid with line plots for BLEU scores. This chart is shown in Fig. 10.2. Aside from the three additional S systems, the scales of the axes are the same as in Fig. 10.1. I then show the values of the top configurations, as before, in Tables 10.6 and 10.7.

10.4 Combined Test Results

In Sections 10.2 and 10.3 I compared configurations from my two systems to the baselines, but I did not compare them to each other, nor did I use test data. In this section, I take the best configuration for comparison from each of LPA and SGA, then compare them to each other and to the baselines over the test data. By *best*, I do not exactly mean the one with the absolute highest intersective BLEU score, which for both LPA and SGA is one of the configurations that combines my transfer rules with those of H&B. Rather, I choose

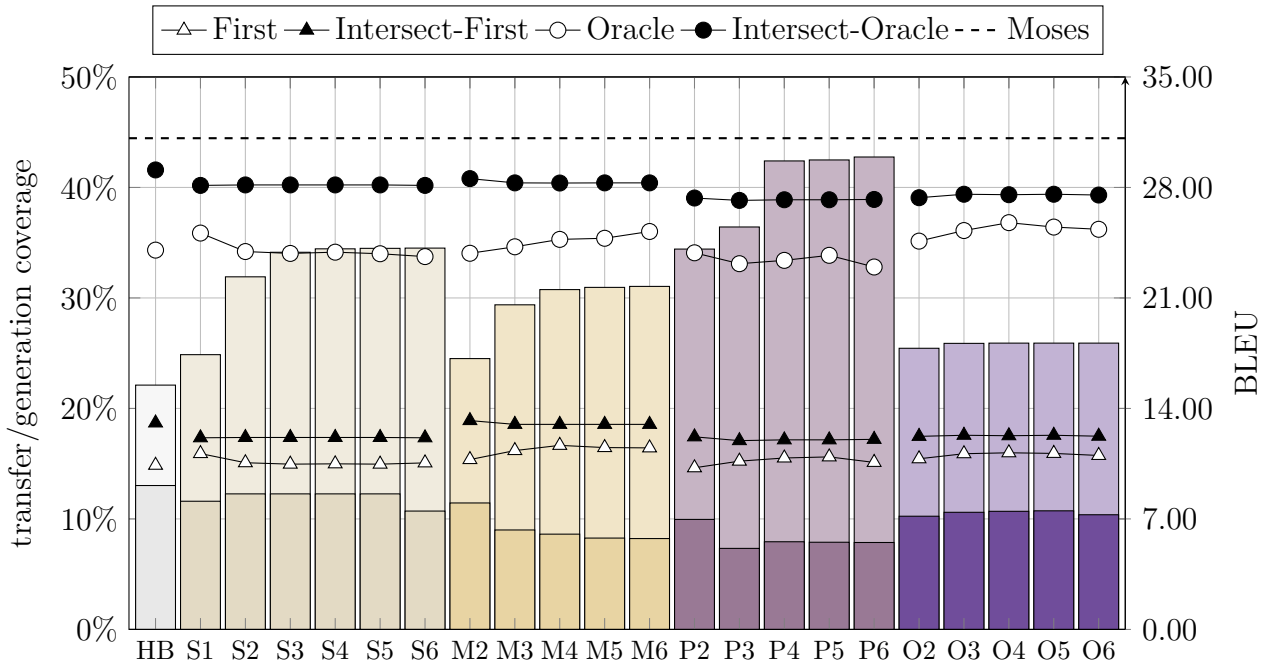


Figure 10.2: Coverage, All-BLEU, and Intersective-BLEU results for all SGA configurations

	% Coverage					
	Transfer		Generation		All-BLEU	
System	Rel	Abs	Rel	Abs	First	Oracle
H&B	27.9	22.1	**58.9**	**13.0**	10.52	24.09
SGA-S2	40.2	31.9	38.4	**12.3**	10.56	23.94
SGA-M2	30.9	24.5	**46.7**	11.4	10.76	23.83
SGA-P2	**43.4**	**34.4**	28.9	10.0	10.24	23.86
SGA-O5	32.6	25.9	41.4	10.7	11.15	25.49

Table 10.6: Coverage and All-BLEU for top SGA configurations

	BLEU		NIST		METEOR	
System	First	Oracle	First	Oracle	First	Oracle
Moses	**31.12**		**5.48**		**32.84**	
H&B	13.08	**29.11**	3.61	**5.22**	26.63	32.44
SGA-S2	12.16	28.16	3.53	5.04	26.34	32.01
SGA-M2	**13.23**	**28.56**	**3.63**	**5.21**	**26.76**	**32.64**
SGA-P2	12.20	27.33	3.52	4.98	26.55	31.89
SGA-O5	12.30	27.57	3.54	5.01	26.50	31.98

Table 10.7: Evaluation results for top SGA configurations over the intersection of development translations

one of the configurations that use only my rules (that is, a P or O configuration), and from those I choose the one with the highest intersective Oracle BLEU scores. The first selected configuration, LPA-O2, is only 0.09 points below the top combined system (LPA-S2), and it beats both baselines. The second, SGA-O5, is 0.66 lower than the top combined system (LPA-M2), and it didn't beat either baseline.

The coverage of the test systems is given in Table 10.8. The transfer coverage of the LPA-O2 system is notable as being the only one to exceed 50% (both relative and absolute). Besides that anomaly, the other two (H&B and SGA-O5) behaved similarly to the development data.

Table 10.9 shows the evaluation scores over the intersection of covered test items. This table contains the second surprise, which is that the SGA system beat the LPA system in all metrics (although it did not beat either baseline in any metric). In addition to the First and Oracle scores for BLEU, NIST, and METEOR, I also asked human annotators to select their preferred translation from the outputs of Moses and SGA-O5. As explained in Section 9.1, the annotators were asked to select the most *useful* translation, not necessarily the most

	% Coverage					
	Transfer		Generation		All-BLEU	
System	Rel	Abs	Rel	Abs	First	Oracle
H&B	27.9	21.8	**57.6**	12.6	9.36	23.97
LPA-O2	**65.8**	**51.5**	35.8	**18.4**	8.27	18.13
SGA-05	31.5	24.6	**43.0**	10.6	10.71	25.56

Table 10.8: Coverage and All-BLEU for testing configurations

fluent, and a third option was allowed for cases where the outputs of both systems were equally good or bad.[7] The results show that the annotators preferred Moses 42.9% of the time and my system 36.5% of the time, with the remaining 20.6% having no preference. In total I had four annotators who are all native speakers of English and proficient in Japanese. Randolph's free-marginal Kappa (Randolph, 2005)[8] gives annotator agreement of 0.50. This value shows some agreement, but it is not high enough to generally be considered reliable. Among the individual annotators, preference for my system ranged from 32.4% to 41.2%, for the Moses system it ranged from 36.3% to 49.0%, and for *no preference* it ranged from 9.8% to 30.4%. Bond et al. (2011) previously showed human evaluation preferring their JaEn-based system 52.8%, compared to 47.3% for Moses,[9] and I suspect that many of the issues causing my system to fare relatively poorly on human annotation is the lack of ranking and post-transfer clean-up that Bond et al. (2011) did (see Section 12.2 for my proposed improvements to my system).

[7]Note that I excluded items where the output of both items was identical.

[8]An alternative to Fleiss' Kappa which improves on the situation where annotators are not required to assign a fixed number of items to certain categories. Values range from -1.0 for perfect disagreement above chance, 0.0 for no agreement above chance, and 1.0 for perfect agreement above chance.

[9]The extra 0.1 is rounding error; they also assigned half-points to both systems when annotators selected *no preference*.

	BLEU		NIST		METEOR		
System	First	Oracle	First	Oracle	First	Oracle	Human
Moses	**35.05**		**5.95**		**35.80**		42.9%
H&B	13.12	**30.40**	3.83	**5.56**	29.74	**35.77**	–
LPA-O2	11.68	24.86	3.48	4.94	28.26	32.72	–
SGA-O5	**12.58**	**29.74**	**3.80**	**5.47**	**29.31**	**34.83**	36.5%

Table 10.9: Evaluation results over the intersection of test translations

10.5 Translation Examples

In this section I will present several examples of translations over the development data. Each table below contains the source Japanese sentence, the reference English sentence, then the translations from Moses, H&B, and various configurations of my two experimental systems. Fig. 10.3 shows the translations from the top configuration in each set of LPA. Fig. 10.4 similarly shows translations from the top configurations of SGA. Fig. 10.5 uses the systems I used for evaluating the test data in Section 10.4. Discussion of these and other results will be given in Chapter 11.

Japanese	ジムは鍵を回した。
Reference	*Jim turned the key in the lock.*
Moses	`Jim pass me the key.`
H&B	`Jim turned the key.`
LPA-S2	`Jim passed on the key to oneself.`
LPA-M2	`Jim turned the key.`
LPA-P2	`Jim passed on the key to myself.`
LPA-O2	`Jim passed on the key to yourselves.`

Figure 10.3: Oracle Translations for the top configurations of LPA

Japanese	彼女は約束を破らない。
Reference	*She is sincere in her promise.*
Moses	`She never breaks his promise.`
H&B	`She does not defeat promise.`
SGA-S2	`She does not break her signing off.`
SGA-M2	`She never breaks promise.`
SGA-P2	`She never breaks her signing off.`
SGA-O2	`She never breaks her signing off.`

Figure 10.4: Oracle Translations for the top configurations of SGA

Japanese	そのネコは茂みの影に隠れていた。
Reference	*The cat lay hidden behind the bushes.*
Moses	`The cat was hiding behind a bush.`
H&B	`Hiding away that cat for the bush shadow.`
LPA-O2	`The cat was hiding at the bush shadow.`
SGA-O5	`The tabby was hiding at the bush shadows.`
Japanese	彼はテレビのニュースキャスターとして働いている。
Reference	*He works as a newscaster in television.*
Moses	`He worked as a tv ニュースキャスター.`
H&B	`He is working as a television anchorman.`
LPA-O2	`He is working as anchormans on a television.`
SGA-O5	`He is working as a television anchorman.`

Figure 10.5: Translations from LPA-O2 and SGA-O5

10.6 Chapter Summary

This chapter presented the results of the experiments described in Chapter 9. Section 10.1 reported the coverage and translation quality estimates for the baseline systems over the development data. Section 10.2 gave an overview of the coverage and quality estimates for all configurations of LPA over the development data, as well as quality estimates over the intersection of the configurations' outputs and those of the baseline systems, which resulted in more comparable scores. Similar numbers and comparisons were reported for SGA in Section 10.3. In Section 10.4 I report the results of running the baselines and the top configurations from LPA and SGA over the test data and provide a comparison between both baselines and both of my systems. Finally in Section 10.5 I provided some example translations for a variety of configurations. In Chapter 11 I will analyze these results and do error analysis over some system outputs.

Chapter 11
ANALYSIS

In this chapter I analyze the results reported in Chapter 10 and perform error analysis. I begin in Section 11.1 by comparing translation outputs across configurations. In Section 11.2 I look into the correlation of automatic quality estimates (e.g., BLEU and METEOR) with transfer and generation coverage (Section 11.2.1), the differences and trends of the quality estimates across configurations (Section 11.2.2), and the stability of the quality estimates over First- and Oracle-selection for LPA and SGA. Section 11.3 investigates the combinatorics and empirical prevalence of possible subgraph topologies and reasons about how they relate to the creation of high-quality transfer rules. In Section 11.4 I examine the relationship between subgraph properties and ill-formed semantic outputs. Section 11.5 looks at the performance of transfer and generation in terms of the amount of duplicate transfers, timeouts, invalid lexical material, and other metrics or problems. In Section 11.6 I examine individual translation outputs and look for the causes of their errors.

11.1 Translation Analysis

Comparing the output of the systems to each other directly can be informative. In this section, I will count how many times the systems gave the same output (i.e., *overlap*) and compare instances where they do not.

Fig. 11.1 shows for LPA (gold) and SGA (purple), for First-selection (lighter shade) and Oracle-selection (darker shade), the overlap[1] with H&B (marked with circles) and each system's most-common translation (MCT; marked with diamonds).[2] The MCT for a system

[1]For Figs. 11.1 and 11.2 I normalize the translations by downcasing and, for Moses, detokenizing and adding . as the default sentence-final punctuation if there is none already.

[2]The most-common translation is system-specific and does not consider the baseline translations.

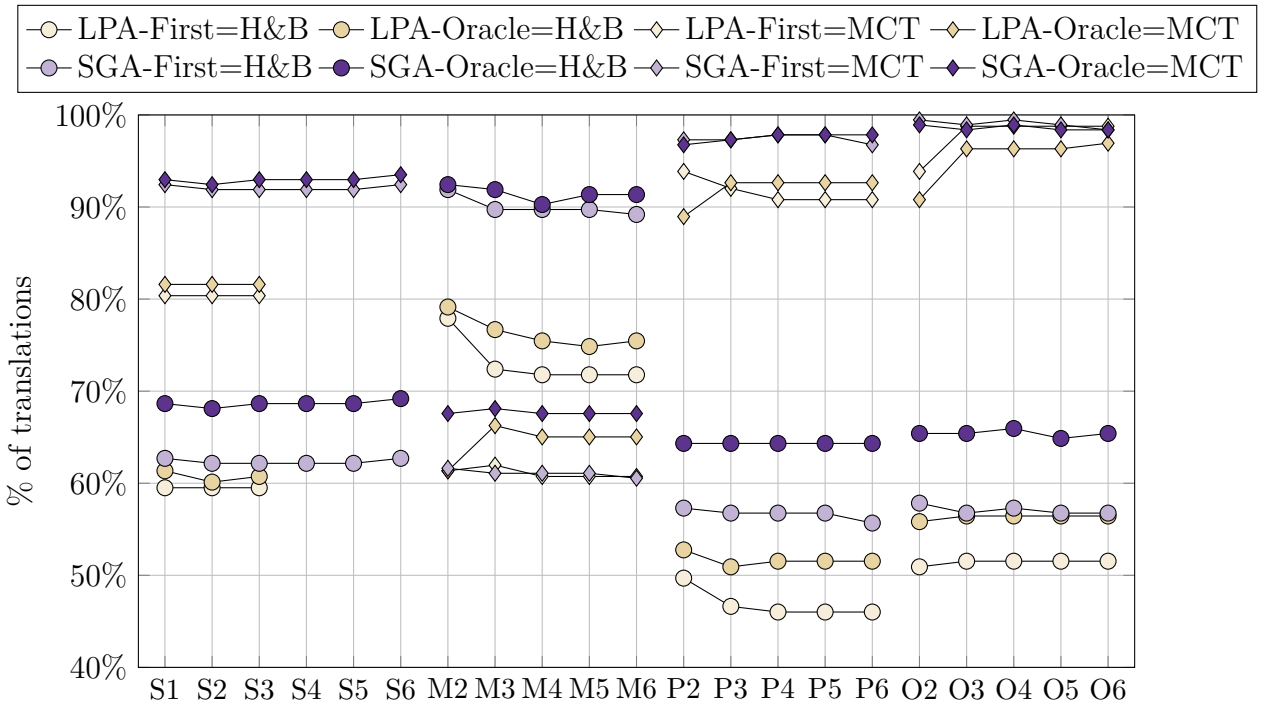


Figure 11.1: LPA and SGA overlap with H&B and the most common translations

(i.e., for LPA or SGA) is the most commonly selected translation for an item across all experimental configurations and this metric is the proportion of items in a configuration that overlap with their MCT. Note the numbers are counted over the intersection of the development data and that the scale begins at 40% to emphasize differences.

An increase of overlap in Fig. 11.1 is interpreted as a decrease of variability. It is neither good nor bad for systems to have high overlap, although such low variability could imply there are untapped MTRs in the transfer grammars or possibly that the grammars themselves are inexpressive. From First to Oracle selection there is a decrease of variability, except for P2 and the O2–O6 set in LPA. First selection is influenced by the relative order of the transfers sent to generation, but Oracle selection sorts all translations according to a single metric—their BLEU score—so it makes sense that Oracle translations have more overlap.

The M and P sets exhibit greater variability than S and O, which is likely due to their

large number of relatively unconstrained MWE rules. In contrast, S's MWE rules (from H&B) are templatic and O's are structurally constrained to be isomorphic. The P and O sets have less overlap with the H&B outputs, as they do not share any H&B rule sets (although it is certainly possible for them to have acquired equivalent rules). Besides the M set, all configurations produce the same translation as most other configurations, both with First and Oracle selection, for at least 80% of their inputs, and for most of those its even over 90%, nearing 100% for the O set. The M set is particular in having more overlap with H&B than with its own MCT, and this is not true for the S set. Overlap with H&B and overlap with MCT are not mutually exclusive—the H&B translation can also be the most common translation for an experimental system—but they are not independent variables. If configurations within a system, such as those in the S, P, and O sets, have high overlap with MCT and relatively lower overlap with H&B, then other configurations in the same system, such as those in the M set, with high H&B overlap would be expected to have relatively lower MCT overlap, as the MCT is the most common across all configurations. Considering that the S, P, and O sets all share my single rules and their outputs look like each other's—and that the M set shares its single rules with H&B and its outputs look more like H&B's—I can conclude that, for the current iteration of my system, the single rules are important for producing translations characteristic of a particular rule extraction methodology.

Fig. 11.2 plots the overlap of LPA outputs on SGA's MCT and vice versa. This chart shows how similar their outputs are to each other, rather than to the H&B baseline or their own MCTs. Mostly it is stable around 70%, but the M set shows significantly less overlap. This finding reinforces the claim that the single rules in my systems have more bearing on the translation outputs than the MWEs. This claim suggests, conversely, that the MWE rules are not very important (in my systems) for translation quality and that perhaps the MWEs are not even being used. Additional tuning of the rule filters and rule ordering can possibly increase the impact of MWE rules on translations but this is left to future work.

The overlap with Moses or with the reference translation is much lower and generally much more consistent, although the S set shows slightly higher and the M set slightly lower

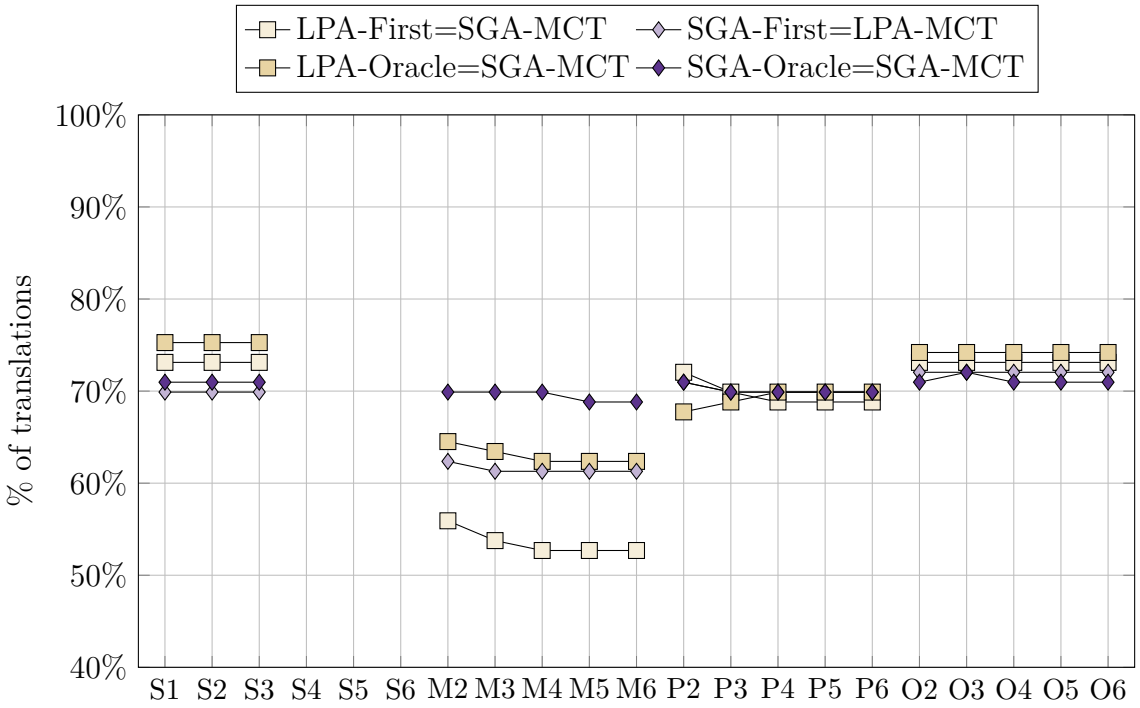


Figure 11.2: Cross overlap between LPA and SGA translations

overlap than other sets. Table 11.1 shows the average overlap across all configurations for each system.

11.2 Comparing Systems by Automatic Quality Estimations

The BLEU (Papineni et al., 2001), NIST (Doddington, 2002), and METEOR (Lavie and Agarwal, 2007) are the three automatic translation quality estimation metrics that I use for my results. Generally, NIST and METEOR agree with BLEU in terms of relative rankings of systems or configurations. NIST disagrees with BLEU in a few cases where the BLEU scores are very close, but METEOR disagrees most strongly in the LPA development data, where the Moses system reports the highest BLEU but the lowest METEOR score. This is not, however, a general trend, as the METEOR score for Moses beats all SGA configurations on development data and for both systems on the test data.

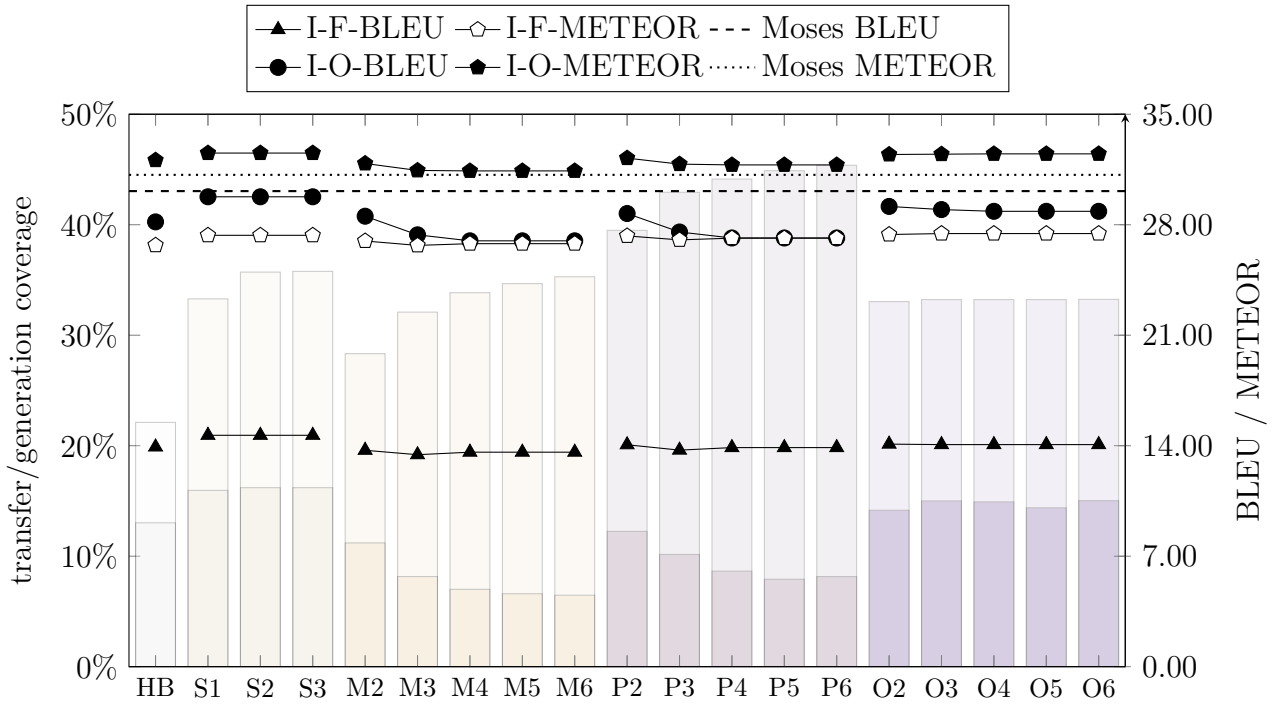
	Condition	
System	= Moses	= Reference
Moses	–	14.1
H&B-First	3.1	4.3
H&B-Oracle	5.5	12.7
LPA-First	2.0	2.8
LPA-Oracle	4.1	10.7
SGA-First	1.1	2.7
SGA-Oracle	5.5	10.5

Table 11.1: System-level overlap with Moses and the reference translation

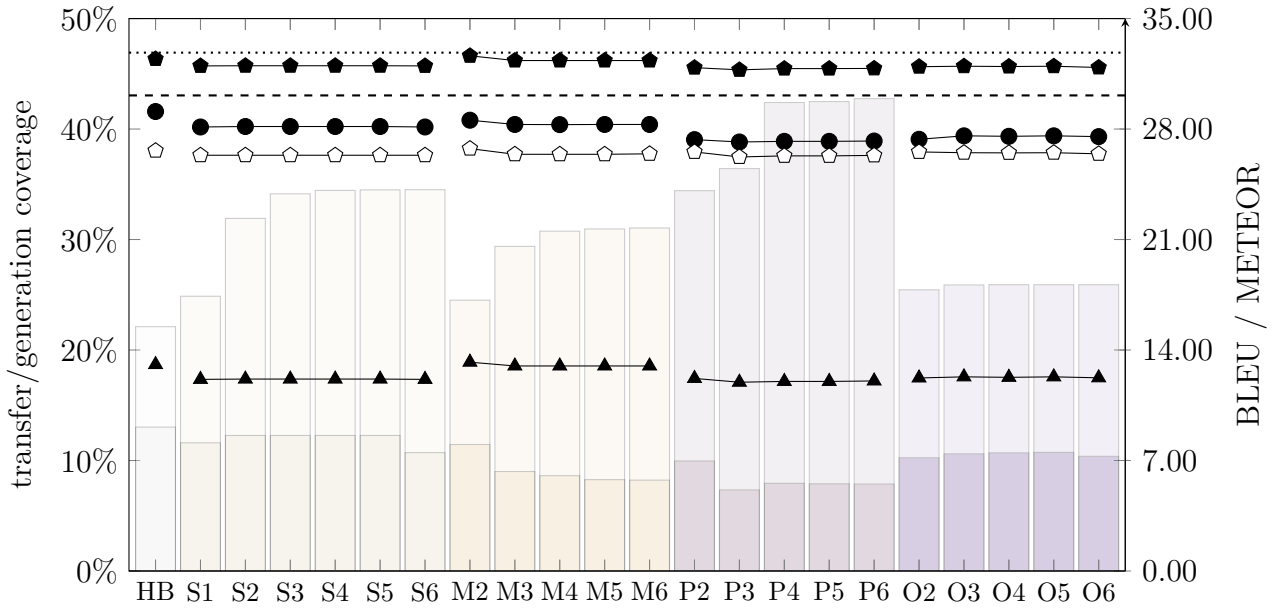
In Figs. 11.3a and 11.3b I repeat the results shown in Figs. 10.1 and 10.2 and add the First-METEOR and Oracle-METEOR scores for comparison. I will cover some of the properties that can be found by examining these charts.

11.2.1 Coverage Versus Quality

One might expect that an increase in the absolute transfer coverage would increase the absolute generation coverage as well but this is generally not the case for my systems, particularly LPA. Indeed, the system with the lowest transfer coverage, the H&B baseline, has one of the higher absolute generation coverages compared to LPA configurations and the highest compared to SGA configurations, and it always has the highest relative generation coverage. In contrast, the configuration with the highest transfer coverage in both systems, P6, has one of the lowest generation coverages. These facts suggest that the systems that make use of non-isomorphic and non-templatic MTRs are transferring MRSs that cannot be realized, viz., because they are not well-formed MRSs or are MRSs not modeled by the target grammar. With the millions of subgraph pairs I'm able to extract (see Sections 9.6.2 and 9.7.2),



(a) LPA



(b) SGA

Figure 11.3: Intersective-First and Intersective-Oracle scores for BLEU and METEOR over all LPA and SGA configurations

it is easy to create very large transfer grammars with multiple MTRs matching the same inputs at various graph orders but without more selective transfer ranking, the list of transfer results fills up with ill-formed MRSs, pushing the good MRSs out of the top five (or top 25, considering that there's potentially five parse results as input to transfer). In other words, the larger grammars seem to have increased recall at the expense of precision.

The other configurations, namely those in the S and O sets, do not exhibit this inverse relationship between absolute transfer and generation coverage. They are more selective about which MTRs are included in a grammar but also there are fewer rules to be added; e.g., for the Tanaka development data subset, LPA-O6 only has 126 more rules than LPA-O5, and SGA-O6 only has 25 more than SGA-05. Given that MTRs of larger graph orders are preferred over smaller ones (see Section 7.2.2), I would expect to see some more variation, good or bad, of coverage or quality. All configuration sets, but particularly the S and O sets, show little to no variation for graph orders 4, 5, and 6 in both LPA and SGA. There are several reasons, possibly more, why adding more rules does not change the coverage or quality:

1. the larger rules can be built compositionally from smaller rules, thus nothing is gained

2. ill-formed MRSs output from larger rules, which would hurt coverage, are also ill-formed from smaller rules, thus there is no change to generability

3. semantic material that would match the input of a rule was partially overwritten by a previous non-optional rule of the same size, thus the second rule is not applied

11.2.2 Relative Scores and Trends

There is a big jump from First selection to Oracle selection for BLEU. This is not surprising, as Oracle is designed to optimize for BLEU and First does not use an n-gram language model to rank outputs, so there is no facility for targeting word orders preferred by the training

corpus. There is a relatively small jump in METEOR scores from First to Oracle selection. The difference in First–Oracle jumps will be discussed further in Section 11.2.3.

There is very little variation either within a set (e.g., all S or all O configurations) and little variation across all configurations. Again, this is not surprising for the Oracle scores, as the systems for each configuration are likely to select similar, or even the same, realized strings for each configuration and the pool of realized sentences can be very large (five parses, five transfers, and twenty realizations means there are potential $5 \times 5 \times 20 = 500$ realizations per input item). More interesting is the lack of variation among First-BLEU scores, which suggests that that the different configurations might be transferring equivalent MRSs. The METEOR scores also show flatter trends than the BLEU scores (again, see Section 11.2.3 for further discussion).

None of my configurations in LPA or SGA beat Moses on BLEU, which is what I expected. All of my LPA configurations beat Moses on METEOR, but they are all very close. For SGA, however, none beat Moses on METEOR, but again it is very close. With such slight differences, the "winner" could swap just by calculating over different subsets of the data (Moses's METEOR score over the LPA intersection is lower than all SGA METEOR scores, while the LPA scores are fairly consistent with those of SGA).

The Oracle BLEU scores for the M and P sets and to a lesser degree the O set for LPA show a downward trend as the graph order increases, which suggests that the MWE MRS transfer rules (MTRs) I create are causing bad transfers to push the good ones off the beam. It's not immediately clear whether this is from MWE rule quantity (according to Table 9.7 on Page 169, the M set ranges from ∼65k–219k rules for the Tanaka development data and the P set from ∼35k–190k, whereas the O set ranges from ∼19k–24k rules), or quality (the M and P sets include non-isomorphic rules, whereas the O set only includes isomorphic rules). In SGA this trend is less severe and it is even reversed for the O set such that O5 has the highest Oracle BLEU. This reduction and reversal of the trend is perhaps due to lower-order pairs being noisier (i.e., having less translationally equivalent mappings) and the higher-order subgraphs intervening before the lower-order ones are applied.

11.2.3 Metric Stability

The difference between Moses and my systems is larger for BLEU than for NIST and METEOR. Here I look at the difference between Moses and all configurations of each system in order to distill down the mean difference and quantify the variance in the metrics. First I rescale each score $s \in S$ such that the highest score becomes 1.0: $s_{norm} = \frac{s}{\max(S \cup \{m\})}$, where m is the Moses score for the corresponding metric. I also calculate the normalized Moses score m_{norm} similarly. I then find the difference between the normalized Moses score and my normalized scores: $\Delta = \{|m_{norm} - s_{norm}| : s_{norm} \in S_{norm}\}$. The mean and standard deviation of these differences are shown in Table 11.2 for both the intersective-First and intersective-Oracle scores for LPA and SGA. The metric with the largest mean difference, as well as the most variance, is BLEU, followed by NIST, then METEOR. The differences are even more pronounced in the systems using First-selection rather than Oracle, with NIST and METEOR showing more stability. These differences highlight the sensitivity of the metrics to matching the reference string, as BLEU looks for exact tokens, NIST gives more weight to rare n-grams and has a more forgiving brevity penalty, and METEOR performs stemming and synonym matching to make the metric even more robust to small differences from the reference translation.

Comparison System	Δ^{BLEU}	Δ^{NIST}	Δ^{METEOR}
LPA First	0.551 ± 0.126	0.296 ± 0.031	0.123 ± 0.009
SGA First	1.439 ± 0.202	0.531 ± 0.043	0.239 ± 0.010
LPA Oracle	0.062 ± 0.035	0.046 ± 0.013	0.028 ± 0.014
SGA Oracle	0.106 ± 0.015	0.079 ± 0.014	0.024 ± 0.007

Table 11.2: Difference in normalized BLEU, NIST, and METEOR between Moses and my systems (mean ± standard deviation)

11.2.4 Summary

I examined the relationship between coverage and translation quality in Section 11.2.1, the relative scores and trends of quality estimations in Section 11.2.2, and looked at the stability of the quality estimation metrics in Section 11.2.3. Through this analysis and that of Section 11.1, the story explaining the systems' performance begins to take shape. MWE rules have the potential to yield more natural and semantically adequate translations but increasing the size and number does not correlate with increases in translation quality. Next I will look further into properties of the subgraph pairs that are the source of information for my transfer rules.

11.3 SGA Subgraph Topologies

The size and shape of the semantic subgraphs are strongly tied to my ability to create useful transfer rules from them. In this section I will look into these properties to reason about challenges with automatically building expressive transfer grammars.

It is much easier to find a full bilingual variable binding (see Section 7.3.2), which I suspect plays a large role in generability of transferred MRSs, when the graphs are isomorphic but as the graph size increases it becomes increasingly unlikely that the two will be isomorphic. Ignoring variable types and argument roles, there are a limited number of structural variations possible. Graphs of order 1 and 2 have only one structure each, assuming graphs must be connected. An order-3 graph has two possible structures (where the two internal nodes are both arguments of the top one, or the third node is an argument of the second and the second of the first.

In general, any internal subgraph structure is possible as long as there is exactly one outer (i.e., top) node. The number of graph structures for graphs of n nodes is the Catalan number for $n-1$, where a Catalan number is defined as $C_n = \frac{1}{n+1}\binom{2n}{n}$. Table 11.3 shows the number of graph structures available for orders up to 6. The examples column shows a shorthand for the structures where, e.g., `(())` could be used for a subgraph like

Order	Number	Examples
1	$C_0 = 1$	`()`
2	$C_1 = 1$	`(())`
3	$C_2 = 2$	`(()()), ((()))`
4	$C_3 = 5$	`(()()()), (()(())), ((())()), ((()())), (((())))`
5	$C_4 = 14$	...
6	$C_5 = 42$	...

Table 11.3: Graph orders and numbers of structural variants

`(x0 / _vessel_n_1 :RSTR-H-of (u0 / _the_q))`. These counts represent the number of abstract structures subgraphs can take but not the variations considered by my isomorphism comparison, which takes into account the variable type and argument roles.

Table 11.4 shows the percentage of order-3 subgraphs in LPA and SGA, from both Jacy and ERG MRSs, that exhibit the given structure. The numbers are taken from subgraph pairs, so the data is already filtered as described in Chapter 6. For LPA, which extracts subgraphs from aligned predicate phrases, the distribution of the two graph structures is about the same. For SGA, which enumerates subgraphs via a rooted traversal, the distributions are nearly opposite, with Jacy producing more non-branching subgraphs and the ERG producing more branching subgraphs. The difference here may be a partial reason for SGA having a relatively larger difference between its isomorphic and non-isomorphic subgraph pairs than does LPA.

Table 11.5 lists the top eight (of more than 300) order-3 structures with roles for Jacy and the ERG in SGA.[3] For Jacy, the most frequent structure is `(:ARG1-EQ-of(:ARG2-NEQ()))`,

[3]Note that one structure in the Jacy column has four roles; this is still an order-3 graph, because two of the roles are re-entrant. This happens with event coordination (e.g., 〆 *te* sentential coordination in Japanese), where the coordinator separately selects the scope handle and event index of the left and right coordinands. In simple sentences where coordinands are not scopally modified, the scope handles and event indices point to the same nodes in DMRS.

Data	% (()())	% ((()))
LPA-Jacy	21.2	78.8
LPA-ERG	24.8	75.2
SGA-Jacy	39.7	60.3
SGA-ERG	68.1	31.9

Table 11.4: Percentage of subgraphs with the given structure

which is used for things like compounding, adnominal $\mathcal{O}$ *no* constructions, and other PP constructions. For the ERG the most frequent structure is (:ARG1-NEQ():ARG2-NEQ()), which is used, e.g., for transitive verbs. The most frequent structure for Jacy is the second most frequent for the ERG and vice versa.

Count	Jacy	Count	ERG
7,521	(:ARG1-NEQ():ARG1-EQ-of())	9,329	(:ARG1-NEQ():ARG1-EQ-of())
7,911	(:ARG2-NEQ(:ARG1-EQ-of()))	10,334	(:ARG1-H():ARG2-H())
7,994	(:ARG2-NEQ():ARG1-EQ-of())	10,638	(:ARG2-NEQ(:RSTR-H-of()))
8,516	(:ARG2-NEQ(:ARG1-HEQ()))	13,828	(:ARG1-NEQ():RSTR-H-of())
14,943	(:L-HNDL-HEQ():L-INDEX-NEQ: R-HNDL-HEQ():R-INDEX-NEQ)	20,702	(:L-INDEX-NEQ():R-INDEX-NEQ())
17,666	(:L-INDEX-NEQ():R-INDEX-NEQ())	25,328	(:ARG1-EQ-of():RSTR-H-of())
19,862	(:ARG1-NEQ():ARG2-NEQ())	29,887	(:ARG1-EQ-of(:ARG2-NEQ()))
80,632	(:ARG1-EQ-of(:ARG2-NEQ()))	33,941	(:ARG1-NEQ():ARG2-NEQ())

Table 11.5: Most frequent order-3 structures, with roles, in Jacy and the ERG

If I include each node's variable type, as is done for isomorphism comparisons, there

are even more variations. It is then no surprise that isomorphism is such a strong filter of subgraph pairs. Not only are there many differences in structure, even for 3×3 pairs, but the distributions of those structures differ between the source and target language as well.

For the rest, i.e., the non-isomorphic pairings, I cannot reliably find a bilingual variable binding for every variable (see Section 7.3.2). A rule without complete variable bindings can still transfer, but the (unbound) semantic material on the output will be inaccessible to successive rules and it will likely result in a disconnected MRS. This situation can explain why configurations with larger graph orders (e.g., M6, P6) have such low generation coverage. I will explore semantic errors (such as disconnectedness) that lead to MRSs that cannot be realized in the next section.

11.4 Semantic Analysis

The connectedness of an MRS (see Section 5.3.1) is one of several well-formedness conditions that the PyDelphin[4] software can test for. Other conditions of well-formedness tested by PyDelphin include: (1) each EP must have a label; (2) each EP must have an intrinsic variable; (3) no EP may have more than one quantifier; (4) every qeq must bind a hole to the label of some set of EPs; etc. I apply these tests to the MRSs output by transfer to see if well-formedness errors correlate with generability.

Fig. 11.4 plots the percentage of transfer outputs that are disconnected (circles) or otherwise ill-formed (triangles) for H&B, LPA, and SGA. As this is over the transfer outputs and not the realizations, there are no separate counts for First or Oracle selection. Since disconnectedness is included with ill-formedness, the former is never higher than the latter.

The H&B set generally has a low number of errors. This is likely due to the fact that the H&B methodology relies on hand-written templates which have the advantage of accurate bilingual variable binding. This in turn avoids the creation of rules that create disconnected structures. I therefore find it surprising that the S set had fewer errors than the H&B set,

[4] `https://github.com/delph-in/pydelphin`

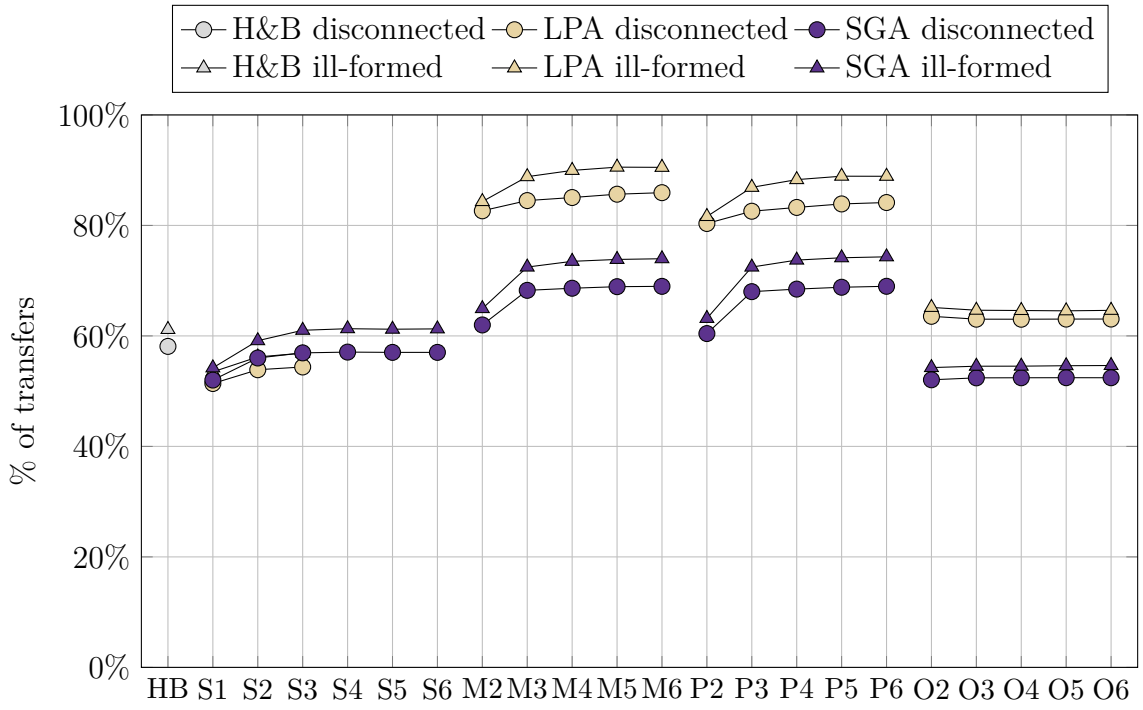


Figure 11.4: Errors in transferred MRSs

although this could be due to outdated templates for H&B's single rules. The chart also confirms my expectation that the M and P sets (with the less-constrained MWE rules) have the most errors and that the O set has fewer than M and P. There is also a noticeable drop (10–20%) in disconnectedness from LPA to SGA outputs, which is perhaps due to the way SGA enumerates subgraphs that are already connected and rooted. That is, the structural constraints in SGA's subgraph enumeration may make it easier for bilingual variable binding to find a full mapping. The O set is peculiar here for not increasing in disconnectedness or ill-formedness as larger rules are introduced, as well as the lowest rates of semantic error overall in SGA, except for S1. It is true that the larger O configurations do not introduce many new rules over their previous iteration, especially for SGA, and this fact by itself could account for the lack of change in semantic errors. There is, however, a relatively large increase in rules from O2 to O3 that does not show an increase in errors, unlike for S1 to S2, M2 to

M3, or P2 to P3. Lastly, the O sets (both for LPA and SGA) do not have many instances of ill-formedness that are not disconnectedness. These facts suggest that the non-isomorphic MWE rules are introducing semantic errors at a higher rate than the isomorphic or templatic rules.

11.5 Performance Comparison

The way the various configurations are constructed may have effects beyond just the coverage and translation quality and these effects may help shed light on problems in the rule selection or transfer grammar augmentation processes. While I only take the top five transfers for my experiments, ACE[5] reports how many it can find before it exhausts the search space, hits a timeout, or hits the memory limit. More specifically, the transfer coverage numbers shown earlier in Figs. 10.1 and 10.2 exclude outputs that include untransferred material, when in fact ACE outputs something—fully or partially transferred—for nearly every input. In contrast, generation in ACE only reports the number of realizations found up to the specified limit and it does not output partial realizations. In the performance charts below, for transfer I use the count of all possible transfers, whereas for generation I use the number of realizations stored.

11.5.1 Transfer Performance

Fig. 11.5 plots the average number of possible transfers per input for each configuration in H&B, LPA, and SGA. The maximum number of transfers for an input is 5,335 and the minimum is zero, but the highest average is around 140. I also plot in the same chart the average number of times a duplicate MRS was transferred per input, i.e., the number of times the same MRS was produced via different paths through the transfer grammar. The number of duplicates reaches as high as 1,296 but usually it is zero, which keeps the average low. The highest average is about 2.7. Note that while the number of duplicates correlates with

[5] `http://sweaglesw.com/linguistics/ace/`

the number of results, the numbers plotted use different scales, with the scale for the number of results on the left and the scale for the number of duplicates on the right of Fig. 11.5. Fig. 11.5 shows the following: (1) LPA has more total results than SGA for the S and O sets; (2) the number of duplicates appears correlated with the number of results for all LPA configurations and for H&B, but not for SGA; and (3) the M set has a relatively low number of results. (1) and (3) above can be explained by looking at when transfer hits the timeout or memory limit.

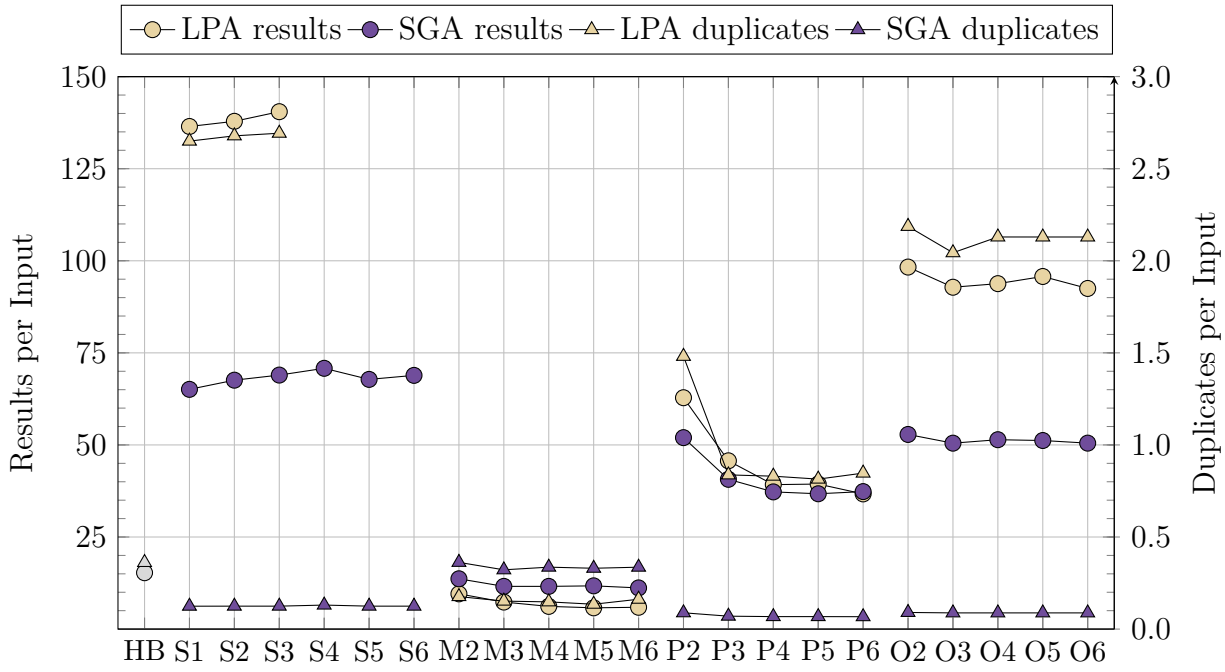


Figure 11.5: Transfer ambiguity in LPA and SGA

Fig. 11.6 plots the average number of times transfer hit a timeout or a memory limit (i.e., a **memout**). While these are both symptoms of an inefficient grammar, the two numbers are mutually exclusive: a transfer task can only hit a timeout or a memout, not both. Both symptoms are relatively infrequent, so I plot both on the same axis, shown on the left. Figs. 11.5 and 11.6 plot different performance metrics but when I scale the charts so the largest value is near the top, the two charts look very similar. Specifically, the number

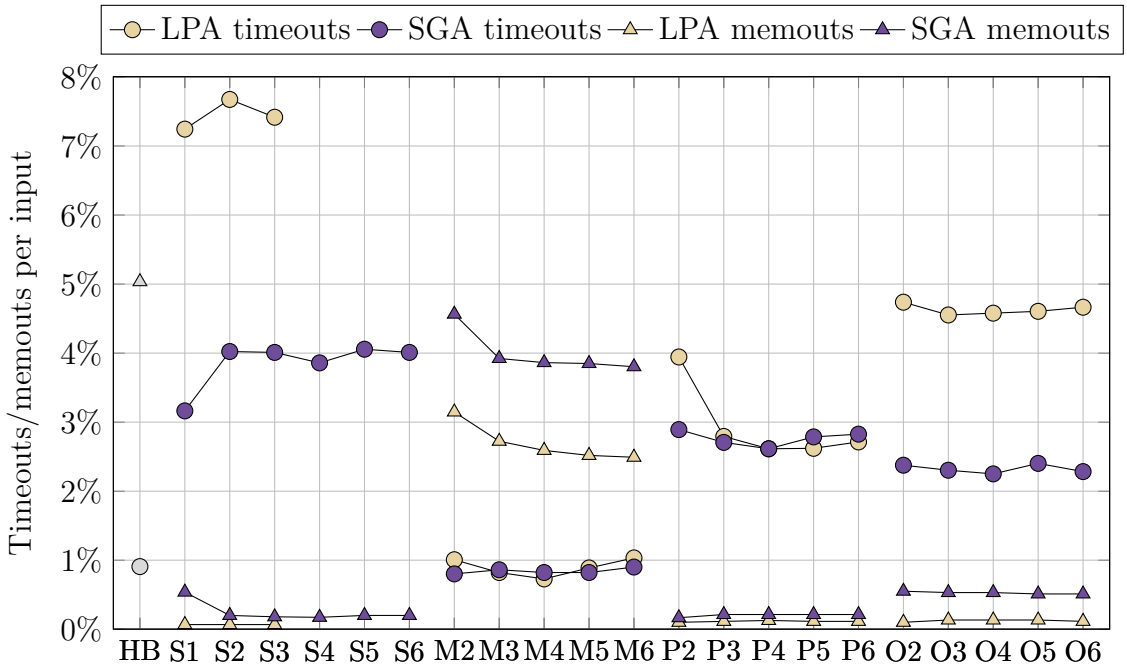


Figure 11.6: Percentage of transfer inputs that hit the timeout or memory limit

of potential transfer results shown in Fig. 11.5 closely follows the frequency of timeouts in Fig. 11.6. This is because, as explained in Footnote 6 on Page 121, transfers that hit the timeout while exploring the search space dump out the partially-transferred MRSs, so these items have many more results than those that completed the search. Those results will likely only be partially transferred and thus not counted in transfer coverage, which explains why there is not a strong correlation between timeouts and transfer coverage (as reported in Figs. 10.1 and 10.2). Memouts, in constrast, cause ACE to abort transfer and return no results, which explains why the M set, which hits memouts more often than other sets, has relatively few results.

As for pattern (2) above—the correlation between duplicates and results for LPA but not SGA, I don't have a good explanation. Perhaps multiple aligned predicate phrases cover the same semantic material but the extracted subgraphs appear different, thus evading my filters for identical subgraph pairs. This situation would lead to a grammar with multiple

transfer rules that do the same thing, thus leading to duplicate MRSs on output. I will not pursue further speculation as to why LPA has more duplicates but I will note that if the duplicate outputs were removed from both LPA and SGA, then SGA would have more potential results than LPA. This would mean that the increase of timeouts in LPA seem to be happening when transfer is exploring search paths that lead to duplicate MRSs.

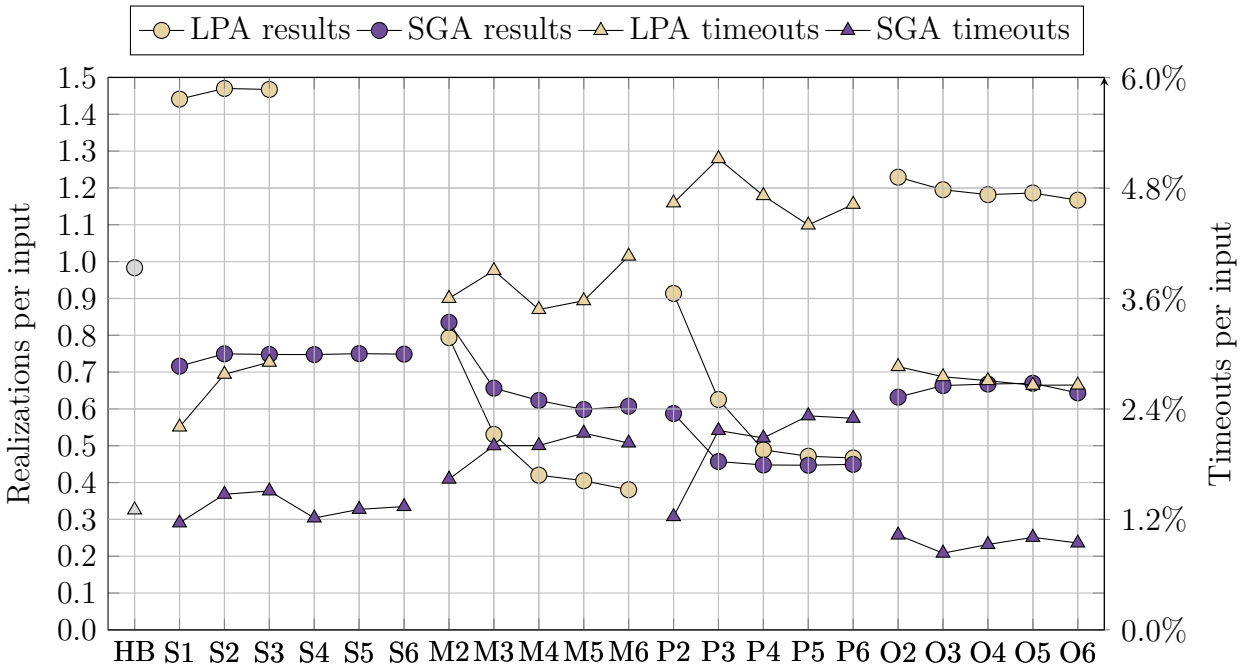


Figure 11.7: Realization ambiguity and timeouts

11.5.2 Generation Performance

For the generation step in the pipeline, Fig. 11.7 shows the number of results per input along with the percentage of inputs that timed out. Fig. 11.8 shows the number of untransferred predicates and transferred predicates that do not match any EP in the target grammar (a **lexical gap**). That is, the both the *untransferred* and *lexical gap* numbers plotted in Fig. 11.8 are the same kind of error for the target grammar—predicates that do not match an EP in the grammar—but one is an untransferred predicate from the source grammar and one is a

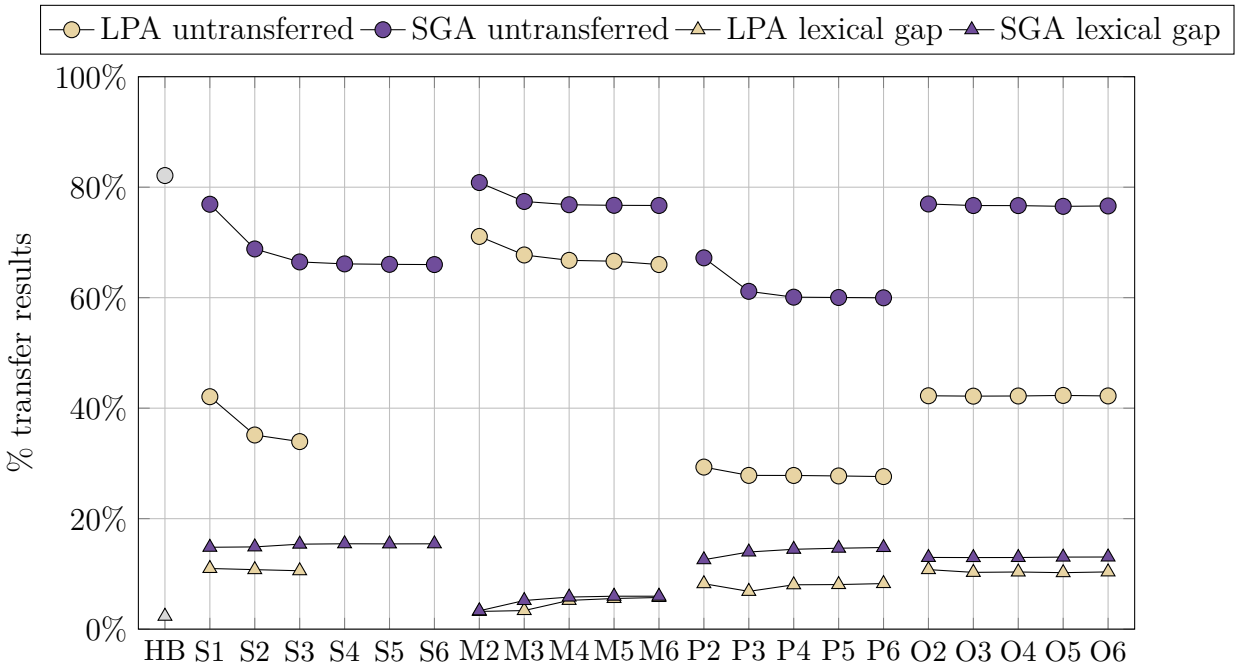


Figure 11.8: Partial transfers and lexical gaps

transferred predicate that simply does not exist in the target grammar (perhaps it is from a rule targeting a previous version of the grammar). Transfer results that have even one predicate that does not map to a target grammar predicate will not result in a realization. Comparing Figs. 11.7 and 11.8, the percentage of inputs with unusable predicates does seem to correlate with the number of realization results, e.g., for the S and O sets, but it does not fully explain, e.g., the M and P sets. When ACE encounters an input with unusable predicates, it terminates the generation process early, so these inputs are unlikely to trigger a timeout. The M and P sets have more timeouts than the other sets, which is probably due to the incomplete bilingual variable bindings in the transfer rules that produced them. Earlier I argued that these incomplete bindings prevent full transfers. These results suggest, however, that even if an input is fully transferred using rules with incomplete bilingual variable bindings, the resulting MRS is underconstrained. Being underconstrained makes it harder for the generator to map the MRS to a derivation, leading to more timeouts.

Considering all three conditions (the two kinds of unusable predicates plus the timeouts), the realization counts in Fig. 11.7 seem to be mostly explained.

11.5.3 Summary

The performance characteristics of the transfer and generation steps are able to largely explain the number of results obtained from each step. For transfer, timeouts, memouts, and the number of duplicate MRSs produced per input explain why LPA produces more transfers in the S and O sets and why the M set has so few transfers. For generation, the number of unusable predicates and timeouts mostly explain why the S and O sets get more realizations in LPA, and why the M and P sets have fewer realizations for the larger rule sizes.

11.6 Translation Examples

Here I present several example translations from configurations of my systems in order to see where they succeed and where they fail. As in Section 10.5, I give the original Japanese sentence, the reference translation, the translation from Moses, then one or more translations from my systems. All of the translations from my systems come from Oracle selection. Often only one translation from my systems is sufficient, because most of them are the same, as discussed in Section 11.1.

One problem I find prevalent concerns resultative or experiential-aspect constructions using -ている *-te iru*, as they have the same form as progressive constructions. Fig. 11.9 gives examples of verbs with experiential aspect marking, where my system translates using the progressive aspect. The first item translates 会っていない *atteinai* "have not met" as *are not meeting*. This example also exhibits a problem with the scope of the negation (*I think you are not* instead of *I don't think you are*)—in fact, if the first problem were fixed (*I think you have not met him*) the output would be a literally correct translation, but in English the phenomenon called **neg-raising** leads to the more idiomatic phrasing exhibited in the reference translation. In Fig. 10.5 in Section 10.5, the second item is similar,

where my system uses the progressive *working* where the habitual *works* is more natural, but in this case it is not a bad translation. The second and third items in Fig. 11.9 exhibit the same problem. In all of these cases, it is likely that the parses given by Jacy have the morphosemantic property for progressive aspect turned on (e.g., if the parse with the progressive reading is ranked higher) and these get passed through transfer to the ERG as I do not handle morphosemantic properties in my transfer rules at all. Even with this error, the output is sometimes an improvement over Moses (as with the second item), and sometimes not (as with the third).

Japanese	あなたは彼に会っていないと思う。
Reference	*I don't believe you've met him.*
Moses	`Have you met him. I don't think.`
SGA-O5	`I think you are not meeting him.`
Japanese	外国の切手を持ってますか。
Reference	*Do you have any foreign stamps?*
Moses	`A foreign stamps?`
SGA-O5	`Are you having foreign stamps?`
Japanese	あなたは彼の兄さんを知っていますか。
Reference	*Do you know his brother?*
Moses	`Do you know his brother?`
SGA-O5	`Are you knowing his brother?`

Figure 11.9: Experiential versus progressive errors in translation

Fig. 11.10 represents a class of errors that mistranslates する *suru* as *do*. する *suru* does mean *do* in many contexts, but it is also used as a light verb, or to mean *to wear*, *to have*, etc. Moses, in contrast, elects to drop the verb altogether, perhaps due to the target language model giving low scores for translations where one is *doing* an expensive necklace.

Idiomatic readings are often translated literally, both by my systems and by Moses. The SGA-S2 translation of the first item in Fig. 11.11 is in fact an accurate literal reading of the Japanese sentence, but 肝をつぶす *kimo o tsubusu* "to smash liver/innards" has the

Japanese	彼女は高価な首飾りをしています。
Reference	*She is wearing an expensive necklace.*
Moses	`She is expensive necklace.`
SGA-O5	`She is doing an expensive necklace.`

Figure 11.10: Wrong sense of する *suru* in translation

idiomatic meaning *to be frightened*. Also, the word つぶす *tsubusu* can mean *to crush*—i.e., *to kill*—in some contexts, which leads to the other mistranslations. For the second item, there are several problems. The first affects mainly Moses and is caused by *X* のために *X-no tame-ni*, which often means *for the purpose of X* or *in order to X*, but here it means *because of X*. The second is a morphological issue affecting my system that leads to a realization of 盗み *nusumi* "stealing" as *steals*.[6] The main error affecting my system, but which Moses got right, is the idiom 首になる *kubi-ni naru* "to be fired", whose surface form literally means *to become* [*a*] *neck*.[7]

Japanese	大きな物音で私は肝をつぶした。
Reference	*The loud noise gave me a terrible fright.*
Moses	`I was in a great noise killed.`
SGA-S2	`I mashed liver with a big noise.`
SGA-M2	`I killed a duct by the big noise.`
Japanese	彼は盗みのために首になった。
Reference	*He was fired for stealing.*
Moses	`He was fired in order to steal.`
SGA-O5	`He became a neck, for steals.`

Figure 11.11: Literal versus idiomatic errors in translation

[6]Both Jacy and the ERG have sophisticated morphological models, but these errors can come up because I do not handle morphosemantic properties, and secondly because the stem form in Japanese (as 盗み *nusumi* is) can be translated in a variety of ways depending on the context.

[7]The idiom evokes the meaning of decapitation as a metaphor for severe punishment, e.g., being dismissed from employment. English has similar idioms, e.g., *to risk one's neck* and *to get it in the neck*.

Fig. 11.12 shows an example where Moses and my system failed to translate a large enough fragment of the input, leading to a collocation error. The literal translation is *He admitted the responsibility of his negligence*, where 自分の過失の責任 *jibun-no kashitsu-no sekinin* "responsibility of one's negligence" is a noun phrase. The reference sentence uses *owned up* to capture the meaning behind a partially overlapping phrase, 責任を認める *sekinin-wo mitomeru* "to admit responsibility". Moses translates just the part about responsibility and ignores the admission, while my systems instead translates the admission and the noun phrase separately, leading to the awkward *his fault responsibilities*.

Japanese	彼は自分の過失の責任を認めた。
Reference	*He owned up to his fault.*
Moses	`He is responsible for his own mistake.`
SGA-S2	`He admitted to his fault blame.`
SGA-M2	`He admitted to his fault responsibilities.`

Figure 11.12: Collocation mismatch error in translation

Note that my translations in Fig. 11.12 would perform better on BLEU because they have the bigram *his fault* where the one from Moses only has unigrams. The better overlap of my translations with the reference string is an effect of the Oracle selection. In other (i.e., unselected) translations, my system outputs different possessive pronouns in place of *his*, including *my*, *her*, etc., but Oracle selection prefers *his* because it appears in the reference string. The fact that my system does not prefer *his* except through Oracle selection brings up another challenge for my systems: pronoun agreement. 彼 *kare* "he" and 自分の *jibun-no* "one's" are close in the surface string, which likely helps Moses choose the appropriate pronoun, but they are very distant in the semantic representation, as shown in Fig. 11.13[8] where the `pron` predicates representing those words are separated by five nodes (or six edges). The distance in the semantic graph makes it difficult for the rules I extract to capture the

[8]The implicit quantifiers are omitted for simplicity.

relationship. Furthermore, neither Jacy nor JaEn utilize any coreference resolution so there is currently no principled way for a reflexive pronoun to agree with its antecedent.

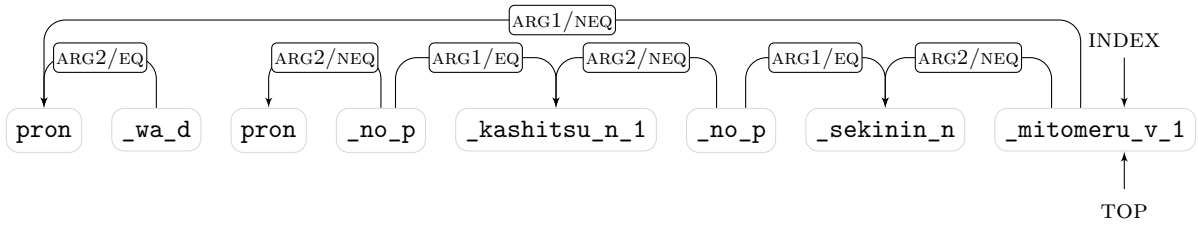


Figure 11.13: DMRS for 彼は自分の過失の責任を認めた

The final example, given in Fig. 11.14, shows a strength of my system. While there are some word-sense mistakes, such as translating 聞く *kiku* "listen" as *hear* and 音楽 *ongaku* "music" as *pieces of music*, my systems correctly translate the negation, which is a crucial part of the meaning. Moses drops the negation entirely, leading to a translation of opposite polarity, although it would perform well on n-gram evaluation metrics as it only differs by one token: *not*. Because the negation occurs on the main verb, the semantic representation has the `neg` predicate on its top node, so it would be unlikely for my systems to ignore it during transfer.

Japanese	彼らは音楽を聞いていませんでした。
Reference	*They were not listening to music.*
Moses	`They were listening to music.`
SGA-O2	`They were not hearing the pieces of music.`

Figure 11.14

11.7 Conclusion

In this chapter I have investigated many aspects of the translation process and its outputs in order to characterize the successes and failures of my systems. In Section 11.1 I looked at

the overlap between my systems, the reference translation, and the baseline translations and found that my systems tend to produce the same translations for the various configurations. In Section 11.2 I found patterns in the translation quality estimation metrics and the coverage which suggest that the larger MWE rules tend to hurt coverage without increasing translation quality. I then investigated why the larger rules are hurting coverage. In Section 11.3 I reasoned that the larger subgraphs, even for 3×3 orders, have many different shapes for the purpose of isomorphism comparisons, which could lead to incomplete variable mappings from the source to the target in the transfer rules. I tested this conjecture in Section 11.4 by analyzing the MRS outputs of transfer for well-formedness, finding that many results of transfer, particularly for the M and P sets, are disconnected MRSs. I then looked at performance characteristics of transfer and generation to see how problems in the semantic representations affected the number of results, timeouts, memouts, etc., and reinforced the finding that the non-isomorphic, non-templatic MWE rules decreased transfer coverage and increased generation timeouts. Finally I looked at a number of actual translation examples to see what kinds of linguistic errors it was making and found that, despite the use of larger semantic fragments in transfer, my systems prefer literal readings of idioms and simplistic morphological mappings.

The main quantitative finding is that I was able to meet and, at times, exceed the results of the H&B baseline without the use of hand-developed, hand-tuned templates. The templates seem to do a better job than my methodology at avoiding the capture of material that is not translationally equivalent and at bilingual variable binding, but it also requires significant manual effort to create the templates and to maintain them with respect to updated versions of the source and target grammars. My method finds many more rules than the templates, especially at larger subgraph sizes, but more work is needed to filter out the bad rules and to more accurately bind the variables bilingually. Once those methods are in place, my method holds the promise of producing more idiomatic rules that accurately transfer semantic fragments and, combined with transfer, realization, and end-to-end ranking as described by Velldal (2008), more natural translations than is currently possible for MRS-

transfer based systems. Furthermore, without the need to manually build templates, my method is more readily applicable to other grammar pairs, i.e., beyond Jacy and the ERG. To be certain, there are settings and processes in my methodology that are built for Jacy to ERG translation, such as the lists of predicates to drop in Tables 9.4 and 9.8 and the SGA subgraph prefilters in Table 9.9, and perhaps implicit biases in my graph traversals and variable binding algorithms. But I kept these distinct as parameters and not hard-coded decisions so that they can easily be swapped out for other values.

In my test results I did not beat the Moses baseline on the automatic translation quality estimates, although the difference is not large, especially for the METEOR metric. I see, however, opportunity in these results, as they were obtained without significant tuning to my system, nor with any sort of ranking of the final transfer outputs. Furthermore, it is simple to obtain a list of predicates that my systems fail to transfer, which helps direct rule extraction efforts. From my analysis above, I think that significant gains could be made by improving the bilingual variable binding for non-isomorphic rules. A simple way of doing this for LPA is to use predicate alignments from Moses instead of Anymalign, as Moses provides internal token (predicate) alignments as well as predicate phrase alignments, and these internal alignments could be projected into the extracted subgraphs. I list some other thoughts on future work in Section 12.2.

Chapter 12

CONCLUSION

This dissertation has presented two novel methods for automatically extracting MRS transfer rules from bilingual semantic corpora in order to augment a transfer grammar for use in machine translation. Semantic transfer grammars, which map representations from one semantic model to another, need frequent maintenance because they lose applicability as the source or target models change (e.g., when the grammars describing the models are updated). Hand-building and maintaining transfer grammars is infeasible because of the need for regular updates and also because the expertise required is immmense—a developer not only needs to be compentent in both the source and target languages, but also needs to be familiar and current with the semantic models of both sides. For these reasons, automatic methods of augmentation are crucial for any practical usage of transfer grammars. My two methods of transfer rule extraction assume a bilingual semantic corpus (or the means to produce one), but otherwise require very little language-specific information in order to extract rules.

My first method, LPA, extends previous work (Haugereid and Bond, 2011, 2012) on repurposing n-gram aligners from phrase-based translation method in order to find alignments of semantic predicate phrases. My second method, SGA, develops a graph-native method of finding semantic subgraph pairs by building a statistical model over enumerated subgraphs—a method inspired by previous methods that traverse semantic (Jellinghaus, 2007) or syntactic (Hearne and Way, 2003; Graham et al., 2009) structures. My methods generally match the performance of the previous methods on automatic evaluation metrics, but do so without the use of hand-built and hand-tuned templates. The methods therefore allow one to bootstrap and maintain a transfer grammar for a pair of existing monolingual

grammars with much less effort than before.

Using LPA, I extracted 358,407 aligned subgraph pairs, which resulted in end-to-end translation coverage of 18.4%. SGA extracted 977,532 aligned subgraph pairs, but had end-to-end coverage of 10.6%, showing that increasing the number of rules does not necessarily lead to higher coverage.[1] On the test data, LPA achieved a BLEU score of 24.86, while SGA achieved 29.74. My two systems trade off coverage for quality, but are generally competitive with the H&B baseline, which had 12.6% coverage and a BLEU score of 30.40. My systems still lag behind Moses (Koehn et al., 2007) on BLEU (i.e., an n-gram based evaluation of an n-gram based system), which gets 35.05, but on METEOR, an alternative metric which is more forgiving of minor variations and has been shown to correlate more strongly with human judgments (Banerjee and Lavie, 2005; Lavie and Agarwal, 2007), my SGA system gets 34.83 compared to Moses's 35.80.

Below I briefly cover some of the methodological and artifactual (i.e., software) contributions of this dissertation in Section 12.1. I reflect on some of my challenges and propose directions for future work in Section 12.2. Finally, I offer some big-picture concluding remarks in Section 12.3.

12.1 Methodological and Artifactual Contributions

This dissertation resulted in several new methods for working with semantic representations and for the transfer rule extraction task. I created a method for projecting a list of predicates onto semantic graphs and enumerating the unique subgraphs that can be built around nodes matching the predicates (see Sections 5.6.1 and 6.1). I devised a test for link orientation that follows the principles of well-formedness for MRS instances in order to perform rooted traversals of DMRS graphs (Sections 5.3.3 and 5.4.2). The rooted traversal yields a singly-rooted DAG, often an arborescence, and often spanning the entire DMRS graph using only the edge directions preferred by the link orientation. The traversal normalizes the source

[1] After filtering to include only task-relevant rules, as described in Section 7.2.1, most configurations in fact have a similar number of rules.

and target graph structures and typically results in nodes being more significant than their descendants to the meaning of the graph overall. I make use of this result to enumerate and pair subgraphs of the traversal as a second method of finding aligned subgraph pairs (Sections 5.6.2 and 6.2), for which I also define a variation of ϕ^2 ranking that discounts the score by the difference in source and target subgraph orders (Section 6.2.4). I also use the graph normalization provided by the rooted traversal with a test for structural isomorphism (Section 5.3.2) to aid in a heuristic for finding bilingual variable bindings, i.e., subgraph-internal node mappings, which help my rules to be more useful in transfer (Section 7.3.2).

In addition to the methodological advancements listed above, the course of this dissertation saw the development of numerous software artifacts as well. PyDelphin,[2] which has already been mentioned in this document, began as a set of functions of MRS representations for the automatic detection of errors in implemented grammars (Goodman and Bond, 2009). For this dissertation I greatly expanded the capabilities of PyDelphin to model DMRS as well as MRS, including functions for serialization, inspection, and transformation; to read TDL (Copestake, 2002) in order to help with grammar inspection; to read and write the YY token format for the robust handling of unknown tokens when parsing with Jacy; to read, write, and transform [incr tsdb()] (Oepen and Flickinger, 1998) profiles, which are heavily used in my translation pipeline; to manage interactions with a running ACE[3] process; and more. PyDelphin is thoroughly documented and follows common software engineering practices, including extensive unit tests, semantic versioning,[4] etc., as it is now used by more people than just me, and it is the backbone of several other projects described below, including: Bottlenose, Demophin, GTest, and XMT. Bottlenose[5] is an HTTP service I wrote, collaborating with Stephan Oepen on the API, that accepts requests for parsing and generating sentences, processes the requests using PyDelphin and ACE on a server, then responds

[2] https://github.com/delph-in/pydelphin

[3] http://sweaglesw.org/linguistics/ace/

[4] https://semver.org/

[5] https://github.com/delph-in/bottlenose

with the results. This service is currently used by two web-based front ends: Demophin and delphin-viz. Demophin,[6] which I created, presents results in a DMRS visualization (which led to the graphical presentation of MRS and DMRS in this document) and allows users to view generation results as well. The other front end, delphin-viz,[7] was created by Ned Letcher and shows derivation trees and MRS representations in addition to DMRS, but does not do generation. Web demonstrations using these front ends and Bottlenose on the back end exist for the ERG and Jacy, both of which proved valuable for inspecting DMRS analyses in my research, and demonstrations also exist for the Mandarin grammar Zhong [|] (Fan et al., 2015), the Indonesian grammar INDRA (Moeljadi et al., 2015), the German grammar GG (Müller and Kasper, 2000; Crysmann, 2005), and the Hausa grammar HaG (Crysmann, 2017). GTest[8] is a grammar debugging tool I wrote that includes semantics well-formedness tests based on those described in Section 5.3. The Penman library[9] is a small package I wrote specifically for serializing and deserializing graphs in PENMAN notation. I use it for reading and writing the bilingually aligned DMRS subgraphs used in this dissertation, but it is equally capable of working with AMR (Banarescu et al., 2013) data. Finally, I created the XMT project[10] which applies the PyDelphin and Penman libraries to create the translation pipeline that I use in my experiments as well as various data-preparation scripts.

Above I listed software projects that started or were significantly developed as a direct result of my research, but the research also resulted in smaller contributions to existing projects. Several bugs in the ERG were found and reported (and often fixed by Dan Flickinger) and many more in Jacy were found and reported (and often fixed by Francis Bond and occasionally by me). I updated, with help from Francis Bond, the JaEn transfer grammar to the most recent releases of Jacy and the ERG. I also updated its rule-extraction tools, which

---

[6] https://github.com/goodmami/demophin

[7] https://github.com/delph-in/delphin-viz

[8] https://github.com/goodmami/gtest

[9] https://github.com/goodmami/penman

[10] https://github.com/goodmami/xmt

were used for the H&B (Haugereid and Bond, 2011, 2012) baseline. My use of ACE for processing led to the discovery of several bugs and deficiences, some of which were fixed by Woodley Packard and others were fixed by me, such as the timeout mechanism for transfer and generation. Finally, I added or expanded documentation on the DELPH-IN wiki,[11] often as a result of discussions within the DELPH-IN community or to aid in my implementation of technologies in PyDelphin.

12.2 Next Steps and Future Research

The exploration of the performance of my two systems revealed a number of possible improvements or directions for future research. In general I could experiment with different thresholds, training data, etc., but I will not discuss those specifically. Some of the improvements below (rule ordering and optionality; rooted enumeration in SGA; automatic post-transfer editing) address deficiencies in my methodology while others (Moses alignments and alternative linearizations for LPA; further DMRS simplifications; EM rule weighting for SGA) are logical next-steps for additional exploration. The future research directions (subgraph interpolation; improved bilingual variable binding; intermediate result ranking and end-to-end ranking; neural parsing, transfer, and generation) represent more significant methodological changes inspired by what I have learned.

Moses Alignments for LPA I only experimented with Anymalign (Lardilleux et al., 2012) alignments of predicate phrases, but Haugereid and Bond (2011, 2012) showed that Moses (Koehn et al., 2007) was able to find alignments that Anymalign could not, so I could potentially increase the number of unique aligned subgraph pairs merely by including Moses alignments. Moses alignments have another potential benefit which has not yet been explored for transfer rule extraction: they include internal (i.e., predicate-to-predicate) alignments which could replace my heuristic bilingual variable binding. These internal alignments are derived from the data, so it is likely they will do better for non-isomorphic subgraph pairs

[11] `http://moin.delph-in.net/`

than my current strategy.

Alternative Linearizations for LPA For LPA I only found n-gram alignments for predicates linearized according to their surface order. Structures built over surface-ordered predicates would look more like syntax than semantics and given that I filter out extracted subgraphs that are disconnected, the extraction process could benefit from a linearization that is closer to the semantic structures. It would therefore be a useful experiment to substitute the surface-order linearization of predicates with a topological linearization, e.g., using the rooted traversal order, and see if it results in more, or better, extracted subgraphs. It may be that such subgraphs would look more like those of SGA, which are enumerated via the same kind of traversal, so the results should be compared for overlap with the SGA subgraphs.

I also limited the n-gram alignment to use the first parse results of the bisem corpus. I have up to five results per item and it is not guaranteed that the first item is the best parse, so LPA could benefit by considering linearizations for parse results beyond the first. Using all five source and target results would increase the size of the training data up to 25 times and many of the additional alignments will be repeated, so it would make sense to simultaneously adjust the frequency or probability thresholds (as described in Section 9.6.2).

Rule Ordering and Optionality A deficiency in the rule selection and ordering process is that a large transfer rule matching the input blocks smaller rules that match a subset of the larger rule's input pattern. This situation is due to (1) larger rules being applied before smaller rules and (2) my rule-optionality strategy where rules are grouped by their input pattern and the last one is made non-optional. Experiments to make all MWE rules optional (so smaller rules get a chance to apply) led to increased timeouts and memouts and ultimately reduced coverage. One possible remedy is to alter the constraint that larger rules always occur first and instead prefer rules where the soure and target subgraphs are of similar order. When smaller rules occur before larger rules and their inputs are a subset of

those of the larger rules, the smaller rules are all made optional. This selective relaxation of rule optionality might be computationally less demanding than making all MWE rules optional, but the task of choosing the appropriate optionality for the rules is significantly more complex. These or other changes to the rule ordering and optionality are applicable for both the LPA and SGA methods.

Further DMRS Simplifications In Section 5.7 I discussed several simplifications of DMRS representations, but noted that I do not convert binary nodes to links as described in Section 5.7.3. This last simplication has the potential to significantly reduce the complexity of the graphs and, as there are relatively few of these nodes occurring frequently, it would only slightly increase the entropy of edge labels. The simplification as described in reversible, so after the subgraph pairs have been extracted I can reify those edges into nodes in order to create the MRS transfer rules. This change, and any other potential graph simplifications, could apply to both LPA and SGA.

Rooted Enumeration in SGA For SGA I use the rooted traversal to get singly-rooted graphs, but I then use a depth-limited traversal for enumeration. The depth-limited traversal includes all nodes at each depth level and was initially created to avoid enumerating too many subgraph variations, but depth alone is not a useful metric for finding subgraphs that capture interesting semantic phenomena. An enumeration based on the rooted traversal might yield more semantically-coherent graphs. For instance, a first pass may extract subgraphs found by traversing only the *to*-oriented links at various depths, which capture the main arguments of a clause. A second pass would then extract order-2 subgraphs including two nodes connected by a *from*-oriented link, which captures basic modification or quantification. If the modifiers or quantifiers themselves have more complicated sub-structures, these should be captured in the first pass. A method such as this would benefit from the binary-node DMRS simplification discussed above.

EM Rule Weighting for SGA Expectation-Maximization (EM) is a learning technique used by word aligners such as Giza++(Och and Ney, 2003) that iteratively converges on an optimal alignment, unlike my weighted-ϕ^2 method for SGA which calculates weights based on a single pass of the data. EM should help avoid more false-positive alignments than my current method.

Subgraph Interpolation One problem in SGA is that the larger subgraph pairs, particularly those involving rare predicates, get assigned the same probability as smaller subgraphs involving the same rare predicates, as the subgraph pairs are equally rare. The assigned probability would be the same whether or not any other less-rare portions of the subgraphs are translationally equivalent. Both LPA and SGA find subgraph pairs of various sizes and many of the smaller subgraphs of the original graph are also subgraphs the larger subgraphs.[12] I can use this fact to interpolate the smaller subgraphs as components of the larger ones, similar to graph composition in graph grammars (e.g., Jones et al., 2012; Groschwitz et al., 2015). Comparisons of the translation probabilities between alternative components, or between components and the larger subgraphs, can possibly help me detect when the source and target are translationally inequivalent.

Improved Bilingual Variable Binding In Chapter 11 I concluded that two of the main issues in my methodology are the shortcomings in bilingual variable binding and the lack of any sophisticated transfer reranker. For the former, the use of Moses alignments (discussed above) could help, particularly for the non-isomorphic subgraph pairs or for linguistic divergences (see Section 1.1) that do not coincide with structural differences in the semantics. SGA cannot make use of Moses alignments, but through subgraph interpolation (see above) it could find the most likely alignments of sub-subgraphs—a method that could also be employed for LPA.

---

[12]For example, a graph has nodes $\{a, b, c\}$, one extracted subgraph has nodes $\{a, b\}$, and another subgraph has just node $\{a\}$.

Automatic Post-transfer Editing I did not incorporate any code to clean up transfer results, although such modifications could yield benefits for translation coverage and quality. For example, all named entities currently need a transfer rule in order to be translated properly, but many could be handled with a more general system that uses an external tool to Romanize Japanese names and otherwise leave the semantics unchanged, aside from stripping the `ja:` prefix. Such a system might detect a node `(x1 / ja:named :carg "坂下")` and output `(x1 / named :carg "Sakashita")`. These changes would be specific to the Japanese–English task and not necessarily generalizable to other language pairs.

Intermediate Result Ranking and End-to-end Ranking My translation pipeline only keeps the top five parse results and transfer results per parse and the top twenty generation results per transfer, relying on the ranking provided by the grammmar and the ACE processor. The ERG's parse ranking is good and its ranking model is updated regularly, but Jacy's ranking model has not been updated in almost a decade, during which time there have been significant changes to the grammar, so it often produces suboptimal parses before the preferred ones. Transfer results are ordered depending on the transfer rule ordering which depends on the size of the input pattern and translation probability associated with the rule (see Section 7.2.2), but there is no ranking of the final transfer results. Generation results are ordered according to the same model as for parsing, but with ACE there is no reranking based on an n-gram language model. The lack of n-gram reranking can hurt the fluency of the outputs and their performance against n-gram metrics such as BLEU. My systems would likely benefit greatly by updating Jacy's parse ranking model and integrating transfer rankers, n-gram realization rankers, and end-to-end rerankers as described by Velldal (2008). Integrating an n-gram realization ranker should also help to bring the results of First selection up closer to the level of those from Oracle selection, which is important for any practical application of the translation system where there is no reference string for comparison.

In addition, I currently have no way of detecting if I arrive at the same (i.e., duplicate) transfer output for different parse results of an item. The same is true of generation results.

My method of selecting the top five or twenty outputs at each step could be improved so that the outputs across branches are compared for duplicates. I store all outputs of each step in a single file (see Section 4.2), so I could initially output more results than I currently do (e.g., 50 transfers per parse result) and subsequently filter them to only keep the first five per parse result that are not duplicated for other parse results of the same item. Alternatively, I could filter them to keep the first N unique results across all transfer results for the same item, effectively implementing a beam search. These strategies would also be relevant and perhaps more helpful for realization results, which currently have many duplicates.

Neural Parsing, Transfer, and Generation The ERG is currently able to generate sentences for most well-formed inputs, but the outputs of transfer are not as clean or as well-formed as the outputs of parsing with the ERG, so currently less than half of the transfer results from my systems are covered in generation. A robust graph-to-string approach would be able to gain the remaining coverage, but the gained items may have lower quality than those that already covered because they are more likely to have ill-formed or incomplete semantic representations. Some recent work applies the Hiero machine translation system (Chiang, 2007) to perform DMRS-to-string transformation both for translation and robust monolingual realization (Horvat et al., 2015; Horvat, 2017). Buys and Blunsom (2017) built a neural encoder-decoder transition-based parser that produces DMRS directly from strings (i.e., without a grammar, although DeepBank (Flickinger et al., 2012), used for training, was produced by a grammar) and Konstas et al. (2017) achieved state-of-the-art results in generating sentences from AMR by applying a seq2seq model based on OpenNMT (Klein et al., 2017) with additions for handling anonymization and AMR syntax. I collaborated with Ioannis Konstas (of Konstas et al. 2017) to adapt their method to Penman-serialized DMRS representations and the initial results were very encouraging. Beyond just gaining coverage, these techniques integrate language models which would also help in getting fluent realizations. The success of Buys and Blunsom (2017) and Konstas et al. (2017) inspires me to consider neural transfer as the last step in creating a robust translation pipeline based

on DMRS. In this case, grammars such as the ERG and Jacy would produce the gold- or silver-standard bisem data that the systems use for training.

12.3 Closing Words

The work I have presented advances an alternative paradigm to machine translation than the purely statistical or neural approaches that are more commonly pursued in recent years, thereby expanding the variety of translation results available. Given that the task of translation is the transformation of source into target sentences with equivalent meaning, the use of a meaning representation rather than surface representations as the medium of that transformation is thus a direct, and intrinsically satisfying, solution to the problem. The transfer of compositional and symbolic semantic representations and the use of precision grammars for analysis and realization makes the translation process very interpretable and inspectable; failures (or successes) in translation can be traced back to specific rules, which can help developers understand how to improve the system. The ability to select alternate parse, transfer, and realization results enables room for customization where ranking models, e.g., trained for specific genres, are available. The LOGON transfer machinery is very powerful and my results were achieved using only a fraction of its capabilities, so improvements to my methodology to target the remaining capabilities (such as rule context or filters), in addition to the improvements discussed above, would lead to a compelling platform for experimentation in semantics-based machine translation.

BIBLIOGRAPHY

Peter Adolphs, Stephan Oepen, Ulrich Callmeier, Berthold Crysmann, Dan Flickinger, and Bernd Kiefer. Some fine points of hybrid natural language parsing. In **LREC**, 2008.

Roee Aharoni and Yoav Goldberg. Towards string-to-tree neural machine translation. In **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**, pages 132–140, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/P17-2021`.

AMR. Abstract Meaning Representation (AMR) 1.2.3 specification, 2017. URL `https://github.com/amrisi/amr-guidelines`. Accessed: 2017-09-06.

R Ananthakrishnan, Pushpak Bhattacharyya, M Sasikumar, and Ritesh M Shah. Some issues in automatic evaluation of English-Hindi MT: more blues for BLEU. **ICON**, 2007.

Anthony Aue, Arul Menezes, Robert Moore, Chris Quirk, and Eric Ringger. Statistical machine translation using labeled semantic dependency graphs. **Proceedings of TMI 2004**, pages 125–134, 2004.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In **Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse**, pages 178–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-2322`.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In **Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization**, pages 65–72, Ann Arbor, Michigan,

June 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W05/W05-0909.

Emily M Bender. On achieving and evaluating language-independence in NLP. **Linguistic Issues in Language Technology**, 6(3):1–26, 2011.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann Copestake. Layers of interpretation: On grammar and compositionality. In **Proceedings of the 11th International Conference on Computational Semantics**, pages 239–249, London, UK, April 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W15-0128.

Rens Bod and Remko Scha. A DOP model for phrase-structure trees. In **Data Oriented Parsing**, Stanford, CA, 2007. CSLI Publications.

Francis Bond, Stephan Oepen, Melanie Siegel, Ann Copestake, and Dan Flickinger. Open source machine translation with delph-in. In **In Proceedings of the Open-Source Machine Translation Workshop at MT Summit X**, pages 15–22, Phuket, Thailand, 2005.

Francis Bond, Hitoshi Isahara, Sanae Fujita, Kiyotaka Uchimoto, Takayuki Kuribayashi, and Kyoko Kanzaki. Enhancing the Japanese WordNet. In **The 7th Workshop on Asian Language Resources**, pages 1–8, Singapore, 2009. ACL-IJCNLP 2009.

Francis Bond, Stephan Oepen, Eric Nichols, Dan Flickinger, Erik Velldal, and Petter Haugereid. Deep open-source machine translation. **Machine Translation**, 25(2): 87–105, 2011.

Johan Bos. **Predicate logic unplugged**. Universität des Saarlandes, 1996.

Johan Bos. Open-domain semantic parsing with Boxer. In **Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA 2015**, pages 301–304, Vilnius, Lithuania, May 2015. Linköping University Electronic Press.

Johan Bos, Yoshiki Mori, Björn Gambäck, Manfred Pinkal, Christian Lieske, and Karsten Worm. Compositional semantics in verbmobil. In **Proceedings of the 16th conference on Computational linguistics (COLING)**, pages 131–136. Association for Computational Linguistics, 1996.

Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. Wide-coverage semantic representations from a CCG parser. In **Proceedings of the**

20th international conference on Computational Linguistics, page 1240. Association for Computational Linguistics, 2004.

Johan Bos, Valerio Basile, Kilian Evang, Noortje Venhuizen, and Johannes Bjerva. The Groningen Meaning Bank. In Nancy Ide and James Pustejovsky, editors, **Handbook of Linguistic Annotation**, volume 2, pages 463–496. Springer, 2017.

Jan Buys and Phil Blunsom. Robust incremental neural semantic graph parsing. In **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pages 1215–1226, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/P17-1112`.

Ulrich Callmeier, Andreas Eisele, Ulrich Schäfer, and Melanie Siegel. The deepthought core architecture framework. In Maria Teresa Lino, Maria Francisca Xavier, Fatima Ferreira, Rute Costa, and Raquel Silva, editors, **Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC) 2004**, pages 1205–1208, Lisbon, Portugal, 5 2004. ELRA, European Language Resources Association.

Gregory Carlson. **Reference to kinds in English**. PhD thesis, University of Massachusetts, Department of Linguistics, 1977.

John Carroll and Stephan Oepen. High efficiency realization for a wide-coverage unification grammar. In **IJCNLP**, pages 165–176. Springer, 2005.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. An efficient chart generator for (semi-) lexicalist grammars. In **Proceedings of the 7th European Workshop on Natural Language Generation**, pages 86–95, 1999.

John Chandioux. MéTéO: un système opérationnel pour la traduction automatique des bulletins météorologiques destinés au grand public. **Meta: Journal des traducteurs/Meta: Translators' Journal**, 21(2):127–133, 1976.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. Improved neural machine translation with a syntax-aware encoder and decoder. In **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pages 1936–1945, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/P17-1177`.

David Chiang. Hierarchical phrase-based translation. **Computational Linguistics**, 33(2):201–228, 2007.

David Chiang. Learning to translate with source and target syntax. In **Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics**, pages 1443–1452. Association for Computational Linguistics, 2010.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. **Syntax, Semantics and Structure in Statistical Translation**, page 103, 2014.

Kenneth W Church and William A Gale. Concordances for parallel text. In **Proceedings of the Seventh Annual Conference of the UW Centre for the New OED and Text Research**, pages 40–62, 1991.

Ann Copestake. **Implementing typed feature structure grammars**, volume 110. CSLI publications Stanford, 2002.

Ann Copestake. Robust Minimal Recursion Semantics. **unpublished draft**, 2004.

Ann Copestake. Semantic composition with (robust) minimal recursion semantics. In **Proceedings of the Workshop on Deep Linguistic Processing**, pages 73–80. Association for Computational Linguistics, 2007.

Ann Copestake. Slacker semantics: why superficiality, dependency and avoidance of commitment can be the right way to go. In **Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics**, pages 1–9. Association for Computational Linguistics, 2009.

Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. Translation using Minimal Recursion Semantics. In **Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation**, pages 15–32, 1995.

Ann Copestake, Alex Lascarides, and Dan Flickinger. An algebra for semantic construction in constraint-based grammars. In **Proceedings of the 39th Annual Meeting on Association for Computational Linguistics**, pages 140–147. Association for Computational Linguistics, 2001.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. Minimal Recursion Semantics. An introduction. **Research on Language and Computation**, 3(4): 281–332, 2005.

Ann A Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszynska. Resources for building applications with Dependency Minimal Recursion Semantics. In **LREC**, 2016.

Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. An improved algorithm for matching large graphs. In **3rd IAPR-TC15 workshop on graph-based representations in pattern recognition**, pages 149–159, 2001.

Berthold Crysmann. Relative clause extraposition in German: An efficient and portable implementation. **Research on Language and Computation**, 3(1):61–82, 2005.

Berthold Crysmann. Reduplication in a computational HPSG of Hausa. **Morphology**, 27(4):527–561, 2017.

Christopher Culy and Susanne Z Riehemann. The limits of n-gram translation evaluation metrics. In **MT Summit IX**, pages 71–78, 2003.

Mary Dalrymple. **Lexical Functional Grammar**, volume 34 of **Syntax and Semantics**. Academic Press, New York, 2001.

Chenchen Ding and Yuki Arase. Dependency tree abstraction for long-distance reordering in statistical machine translation. In **Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics**, pages 424–433, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/E14-1045`.

Yuan Ding and Martha Palmer. Machine translation using probabilistic synchronous dependency insertion grammars. In **Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics**, pages 541–548. Association for Computational Linguistics, 2005.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In **Proceedings of the second international conference on Human Language Technology Research**, pages 138–145. Morgan Kaufmann Publishers Inc., 2002.

Bonnie J Dorr. Machine translation divergences: A formal description and proposed solution. **Computational Linguistics**, 20(4):597–633, 1994.

Rebecca Dridan and Stephan Oepen. Parser evaluation using Elementary Dependency Matching. In **Proceedings of the 12th International Conference on Parsing Technologies**, pages 225–230. Association for Computational Linguistics, 2011.

Rebecca Dridan and Stephan Oepen. Tokenization: Returning to a long solved problem —a survey, contrastive experiment, recommendations, and toolkit —. In **Proceedings**

of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 378–382, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P12-2074`.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In **Proceedings of the ACL 2010 System Demonstrations**, pages 7–12. Association for Computational Linguistics, 2010.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. A simple, fast, and effective reparameterization of IBM Model 2. In **Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**, pages 644–648, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N13-1073`.

Helge Dyvik, Miriam Butt, and Tracy Holloway King. The universality of f-structure: discovery or stipulation? the case of modals. **image**, 2:3, 1999.

Jason Eisner. Learning non-isomorphic tree mappings for machine translation. In **Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 2**, pages 205–208. Association for Computational Linguistics, 2003.

Katherine Everitt, Christopher Lim, Oren Etzioni, Jonathan Pool, Susan Colowick, and Stephen Soderland. Evaluating lemmatic communication. **Journal of Translation and Technical Communication Research**, 3:70–84, 2010.

Zhenzhen Fan, Sanghoun Song, and Francis Bond. An HPSG-based shared-grammar for the chinese languages: Zhong [|]. In **Proceedings of the Grammar Engineering Across Frameworks (GEAF) 2015 Workshop**, pages 17–24, Beijing, China, July 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W15-3303`.

Christiane Fellbaum. **WordNet**. Wiley Online Library, 1998.

Dan Flickinger. On building a more efficient grammar by exploiting types. **Natural Language Engineering**, 6(1):15–28, 2000.

Dan Flickinger, Alexander Koller, and Stafan Thater. A new well-formedness criterion for semantics debugging. In Stefan Müller, editor, **The Proceedings of the 12th International Conference on Head-Driven Phrase Structure Grammar, Department of Informatics, University of Lisbon**, pages 129–142, Stanford, 2005a. CSLI Publications. URL `http://cslipublications.stanford.edu/HPSG/6/`.

Dan Flickinger, Jan Tore Lønning, Helge Dyvik, Stephan Oepen, and Francis Bond. SEMI rational MT-enriching deep grammars with a semantic interface for scalable machine translation. In **In Proceedings of the 10th Machine Translation Summit**, pages 165–172. Citeseer, 2005b.

Dan Flickinger, Yi Zhang, and Valia Kordoni. DeepBank. a dynamically annotated treebank of the wall street journal. In **Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories**, pages 85–96, 2012.

Ruth Fuchss, Alexander Koller, Joachim Niehren, and Stefan Thater. Minimal Recursion Semantics as dominance constraints: Translation, evaluation, and analysis. In **Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics**, page 247. Association for Computational Linguistics, 2004.

Sanae Fujita, Takaaki Tanaka, Francis Bond, and Hiromi Nakaiwa. An implemented description of Japanese: The Lexeed dictionary and the Hinoki treebank. In **Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions**, pages 65–68, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1225403.1225420. URL `http://www.aclweb.org/anthology/P06-4017`.

Daniel Gildea. Loosely tree-based alignment for machine translation. In **Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1**, pages 80–87. Association for Computational Linguistics, 2003.

Michael Wayne Goodman and Francis Bond. Using generation for grammar analysis and error detection. In **Proceedings of the ACL-IJCNLP 2009 conference short papers**, pages 109–112. Association for Computational Linguistics, 2009.

Yvette Graham. **Deep Syntax in Statistical Machine Translation**. PhD thesis, Dublin City University, 2011.

Yvette Graham, Josef van Genabith, Anton Bryl, Miriam Butt, and Tracy Holloway King. F-structure transfer-based statistical machine translation. **Proceedings of LFG09**, pages 317–337, 2009.

Jonas Groschwitz, Alexander Koller, and Christoph Teichmann. Graph parsing with s-graph grammars. In **Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**, volume 1, pages 1481–1490, 2015.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In **Proceedings of the 7th Python in Science Conference (SciPy2008)**, pages 11–15, Pasadena, CA USA, August 2008.

Petter Haugereid and Francis Bond. Extracting transfer rules for multiword expressions from parallel corpora. In **Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World**, pages 92–100, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W11-0814`.

Petter Haugereid and Francis Bond. Extracting semantic transfer rules from parallel corpora with smt phrase aligners. In **Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation**, pages 67–75, Jeju, Republic of Korea, July 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W12-4208`.

Mary Hearne and Andy Way. Seeing the wood for the trees: Data-Oriented Translation. In **In Machine Translation Summit IX**. Citeseer, 2003.

Matic Horvat. Hierarchical statistical semantic translation and realization. Technical Report UCAM-CL-TR-913, University of Cambridge, Computer Laboratory, October 2017. URL `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-913.pdf`.

Matic Horvat, Ann Copestake, and Bill Byrne. Hierarchical statistical semantic realization for minimal recursion semantics. In **Proceedings of the 11th International Conference on Computational Semantics**, pages 107–117, 2015.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. Who did what to whom? a contrastive study of syntacto-semantic dependencies. In **Proceedings of the Sixth Linguistic Annotation Workshop**, pages 2–11, Jeju, Republic of Korea, July 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W12-3602`.

Michael Jellinghaus. Automatic acquisition of semantic transfer rules for machine translation. Master's thesis, Universität des Saarlandes, 2007.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. Semantics-based machine translation with hyperedge replacement grammars. In **Proceedings of COLING 2012**, pages 1359–1376, 2012.

Hans Kamp, Josef Van Genabith, and Uwe Reyle. Discourse Representation Theory. In **Handbook of philosophical logic**, pages 125–394. Springer, 2011.

Robert T Kasper. A flexible interface for linking applications to penman's sentence generator. In **Proceedings of the workshop on Speech and Natural Language**, pages 153–158. Association for Computational Linguistics, 1989.

Karin Kipper-Schuler. **VerbNet: A broad-coverage, comprehensive verb lexicon**. PhD thesis, University of Pennsylvania, 2005.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In **Proc. ACL**, 2017. doi: 10.18653/v1/P17-4012. URL `https://doi.org/10.18653/v1/P17-4012`.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. Abstract Meaning Representation (AMR) annotation release 2.0 ldc2017t10, 2017. Web Download.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In **Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions**, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P07-2045`.

Alexander Koller and Stefan Thater. Computing weakest readings. In **Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics**, pages 30–39. Association for Computational Linguistics, 2010. URL `http://aclanthology.coli.uni-saarland.de/pdf/P/P10/P10-1004.pdf`.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. Neural amr: Sequence-to-sequence models for parsing and generation. In **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pages 146–157, Vancouver, Canada, July 2017.

Association for Computational Linguistics. URL http://aclweb.org/anthology/P17-1014.

Milen Kouylekov and Stephan Oepen. RDF triple stores and a custom SPARQL front-end for indexing and searching (very) large semantic networks. In **Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations**, pages 90–94, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL http://www.aclweb.org/anthology/C14-2020.

Hans-Ulrich Krieger and Ulrich Schäfer. TDL: a type description language for constraint-based grammars. In **Proceedings of the 15th conference on Computational linguistics**, volume 2, pages 893–899. Association for Computational Linguistics, 1994.

Taku Kudo. MeCab: Yet another part-of-speech and morphological analyzer. **http://mecab.sourceforge.net/**, 2005.

Irene Langkilde and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In **Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1**, pages 704–710. Association for Computational Linguistics, 1998.

Adrien Lardilleux, François Yvon, and Yves Lepage. Hierarchical sub-sentential alignment with Anymalign. In **16th annual conference of the European Association for Machine Translation (EAMT 2012)**, pages 279–286, 2012.

Alon Lavie and Abhaya Agarwal. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In **Proceedings of the Second Workshop on Statistical Machine Translation**, pages 228–231, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W07/W07-0234.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. **Transactions of the Association for Computational Linguistics**, 5:365–378, 2017.

Philip M Lewis II and Richard Edwin Stearns. Syntax-directed transduction. **Journal of the ACM (JACM)**, 15(3):465–488, 1968.

Jan Tore Lønning and Stephan Oepen. Taking (out) scope: On the history of scope underspecification and its use to date. Unpublished work presented at the 12th DELPH-IN Summit, retrieved 13 March 2018, 2016. URL `http://www.delph-in.net/2016/eds.pdf`.

Jan Tore Lønning, Stephan Oepen, Dorothee Beermann, Lars Hellan, John Carroll, Helge Dyvik, Dan Flickinger, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, et al. LOGON. a norwegian MT effort. In **Proceedings of the Workshop in Recent Advances in Scandinavian Machine Translation**, volume 6, 2004.

Deryle W Lonsdale, Alexander Franz, and John RR Leavitt. Large-scale machine translation: An interlingua approach. In **Iea/aie**, pages 525–530, 1994.

Adam Lopez. Statistical machine translation. **ACM Computing Surveys (CSUR)**, 40(3):8, 2008.

Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**, pages 1412–1421, 2015.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The Penn Treebank. **Computational Linguistics**, 19: 313–330, 1993.

David Moeljadi, Francis Bond, and Sanghoun Song. Building an HPSG-based indonesian resource grammar (INDRA). In **Proceedings of the Grammar Engineering Across Frameworks (GEAF) 2015 Workshop**, pages 9–16, Beijing, China, July 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W15-3302`.

Stefan Müller and Walter Kasper. HPSG analysis of German. In Wolfgang Wahlster, editor, **Verbmobil: Foundations of Speech-to-Speech Translation**, pages 238–253. Springer, Berlin, 2000.

Makoto Nagao. A framework of a mechanical translation between Japanese and English by analogy principle. **Artificial and human intelligence**, pages 351–354, 1984.

Eric Nichols, Francis Bond, Darren Scott Appling, and Yuji Matsumoto. Combining resources for open source machine translation. In **The 11th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-07)**, pages 134–142, Skövde, 2007.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. **Computational linguistics**, 29(1):19–51, 2003.

Stephan Oepen. The transfer formalism: General purpose mrs-rewriting. Technical report, Department of Informatics, University of Oslo, November 2008. URL `http://www.emmtee.net/reports/11.pdf`. LOGON Technical Report #2007-11.

Stephan Oepen and Daniel P Flickinger. Towards systematic grammar profiling. test suite technology 10 years after. **Computer Speech & Language**, 12(4):411–435, 1998.

Stephan Oepen and Jan Tore Lønning. Discriminant-based MRS banking. In **Proceedings of the 5th International Conference on Language Resources and Evaluation**, pages 1250–1255, 2006.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christoper D. Manning. LinGO Redwoods: A rich and dynamic treebank for HPSG. In **Proceedings of The First Workshop on Treebanks and Linguistic Theories (TLT2002)**, Sozopol, Bulgaria, 2002.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D Manning. Lingo redwoods. **Research on Language and Computation**, 2(4):575–596, 2004.

Stephan Oepen, Erik Velldal, Jan Tore Lønning, Paul Meurer, Victoria Rosén, and Dan Flickinger. Towards hybrid quality-oriented machine translation. on linguistics and probabilities in MT. In **In Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation**. Citeseer, 2007.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In **SemEval@COLING**, 2014.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresova. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In **Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)**, pages 915–926, 2015.

Woodley Packard, Emily M. Bender, Jonathon Read, Stephan Oepen, and Rebecca Dridan. Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem. In **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pages 69–78, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P14-1007`.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. **Computational linguistics**, 31(1):71–106, 2005.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. Technical Report RC22176(W0109-022), IBM Research Report, September 17 2001. URL `http://www.mt-archive.info/IBM-2001-Papineni.pdf`.

Terence Parsons. **Events in the Semantics of English**, volume 5. Cambridge, Ma: MIT Press, 1990.

Penman. **The Penman Documentation and User Guide**. The PENMAN Project, Marina del Ray, California, 1989.

Carl J. Pollard and Ivan A. Sag. **Head-Driven Phrase Structure Grammar**. University of Chicago Press, Chicago, 1994.

Matt Post, Juri Ganitkevitch, Luke Orland, Jonathan Weese, Yuan Cao, and Chris Callison-Burch. Joshua 5.0: Sparser, better, faster, server. In **Proceedings of the Eighth Workshop on Statistical Machine Translation**, pages 206–212, 2013.

Sameer S Pradhan, Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. OntoNotes: A unified relational semantic representation. **International Journal of Semantic Computing**, 1(04):405–419, 2007.

Chris Quirk, Arul Menezes, and Colin Cherry. Dependency treelet translation: Syntactically informed phrasal SMT. In **Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics**, pages 271–279. Association for Computational Linguistics, 2005.

Christopher Quirk and Arul Menezes. Dependency treelet translation: the convergence of statistical and example-based machine-translation? **Machine Translation**, 20(1): 43–65, 2006.

Justus J Randolph. Free-marginal multirater kappa (multirater k_{free}): An alternative to fleiss' fixed-marginal multirater kappa. **Online submission**, 2005.

Uwe Reyle. Dealing with ambiguities by underspecification: Construction, representation and deduction. **Journal of Semantics**, 10(2):123–179, 1993.

Sergio Roa, Valia Kordoni, and Yi Zhang. Mapping between compositional semantic representations and lexical semantic resources: Towards accurate deep semantic parsing. In **Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers**, pages 189–192. Association for Computational Linguistics, 2008.

Victoria Rosén, Paul Meurer, Koenraad De Smedt, Miriam Butt, and Tracy Holloway King. Constructing a parsed corpus with a large LFG grammar. In **Proceedings of LFG'05**, pages 371–387. Citeseer, 2005.

Satoshi Sato and Makoto Nagao. Toward memory-based translation. In **Proceedings of the 13th conference on Computational linguistics-Volume 3**, pages 247–252. Association for Computational Linguistics, 1990.

Ulrich Schäfer. **Integrating Deep and Shallow Natural Language Processing Components – Representations and Hybrid Architectures**. PhD thesis, Faculty of Mathematics and Computer Science, Saarland University, Saarbrücken, Germany, 2007. URL `http://www.dfki.de/~{}uschaefer/diss/`. Doctoral Dissertation; also available as Vol. 22 of the Saarbrücken Dissertations in Computational Linguistics and Language Technology series (http://www.dfki.de/lt/diss), ISBN 978-3-933218-21-6.

David Schlangen and Alex Lascarides. Resolving fragments using discourse information. In **Proceedings of the 6th International Workshop on Formal Semantics and Pragmatics of Dialogue (EDILOG 2002)**, 2002.

Melanie Siegel, Emily M. Bender, and Francis Bond. **Jacy: An Implemented Grammar of Japanese**. CSLI Studies in Computational Linguistics. CSLI Publications, Stanford, November 2016. ISBN 9781684000180.

Stephen Soderland, Christopher Lim, Mausam, Bo Qin, Oren Etzioni, and Jonathan Pool. Lemmatic machine translation. In **Proceedings of the Machine Translation Summit XII**, 2009.

Sanghoun Song. Information structure of relative clauses in english: a flexible and computationally tractable model. **Language and Information**, 18(2), 2014.

Sanghoun Song and Emily M Bender. Using information structure to improve transfer-based MT. In **Proceedings of the 18th International Conference on Head-Driven Phrase Structure Grammar**, pages 348–368, 2011.

Mark Steedman and Jason Baldridge. Combinatory Categorial Grammar. **Non-Transformational Syntax: Formal and explicit models of grammar**, pages 181–224, 2011.

Eiichiro Sumita, Hitoshi Iida, and Hideo Kohyama. Translating with examples: a new approach to machine translation. In **The Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language**, pages 203–212, 1990.

Yasuhito Tanaka. Compilation of a multilingual parallel corpus. In **Proceedings of PACLING 2001**, pages 265–268, 2001.

Kristina Toutanova, Christopher D Manning, Dan Flickinger, and Stephan Oepen. Stochastic HPSG parse disambiguation using the redwoods corpus. **Research on Language & Computation**, 3(1):83–105, 2005.

Bernard Vauquois. Structures profondes et traduction automatique. le système du CETA. **Revue Roumaine de linguistique**, 1968.

Erik Velldal. **Empirical Realization Ranking**. PhD thesis, University of Oslo, Department of Informatics, 2008.

Wolfgang Wahlster. Verbmobil. In **Grundlagen und anwendungen der künstlichen intelligenz**, pages 393–402. Springer, 1993.

Wolfgang Wahlster. Mobile speech-to-speech translation of spontaneous dialogs: an overview of the final Verbmobil system. In **Verbmobil: Foundations of speech-to-speech translation**, pages 3–21. Springer, 2000.

Andy Way. A hybrid architecture for robust MT using LFG-DOP. **Journal of Experimental & Theoretical Artificial Intelligence**, 11(3):441–471, 1999.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. **Computational Linguistics**, 23(3):377–403, 1997.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. Sequence-to-dependency neural machine translation. In **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pages 698–707, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/P17-1065`.

Nianwen Xue, Ondrej Bojar, Jan Hajic, Martha Palmer, Zdenka Uresova, and Xiuhong Zhang. Not an interlingua, but close: Comparison of English AMRs to Chinese and Czech. In **LREC**, volume 14, pages 1765–1772. Reykjavik, Iceland, 2014.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In **Proceedings of the 39th Annual Meeting on Association for Computational Linguistics**, pages 523–530. Association for Computational Linguistics, 2001.

Xiaocheng Yin, Jung-Jae Kim, Zinaida Pozen, and Francis Bond. Parse ranking with semantic dependencies and WordNet. In **Proceedings of the Seventh Global Wordnet Conference**, pages 186–193, 2014.

Yi Zhang, Stephan Oepen, and John Carroll. Efficiency in unification-based n-best parsing. In **Trends in Parsing Technology**, pages 223–241. Springer, 2010.

Ying Zhang, Stephan Vogel, and Alex Waibel. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? In **LREC**, 2004.

Appendix A

SETTING UP THE TRANSLATION ENVIRONMENT

In this appendix I detail the configurations of two parts of my system. In Appendix A.1 I list the experimental configuration I used for my baseline Moses system. In Appendix A.2 I list the customizations I made for the `1214` version of the ERG. All other configuration for my experiments is detailed at the XMT project[1] at:
`https://github.com/goodmami/xmt/tree/master/doc`.

A.1 Moses

Moses[2] is available at `https://github.com/moses-smt/mosesdecoder`. Below I list the script I use with Moses 4.0's Experiment Management System (EMS).

```
[GENERAL]
# environment directories (set as appropriate)
working-dir = /projects/moses-baseline
moses-dir = /NLP_TOOLS/mt_tools/moses/v4.0
external-bin-dir = /NLP_TOOLS/mt_tools/mgizapp/latest/bin
# data directories
input-extension = jp
output-extension = en
pair-extension = jp-en
bitext-dir = $working-dir/bitext
# moses directories
moses-src-dir = $moses-dir
moses-bin-dir = $moses-dir/bin
moses-script-dir = $moses-dir/scripts
decoder = $moses-bin-dir/moses
ttable-binarizer = "$moses-bin-dir/CreateOnDiskPt 1 1 4 100 2"
output-tokenizer = "$moses-script-dir/tokenizer/tokenizer.perl -a
                    -l $output-extension"
output-truecaser = $moses-script-dir/recaser/truecase.perl
```

[1] `https://github.com/goodmami/xmt`

[2] `http://www.statmt.org/moses`

```
detruecaser = $moses-script-dir/recaser/detruecase.perl

[CORPUS]
max-sentence-length = 70

[CORPUS:kw-tc-wn]
raw-stem = $bitext-dir/train

[LM]
# kenlm training
lm-training = "$moses-script-dir/ems/support/lmplz-wrapper.perl
               -bin $moses-bin-dir/lmplz"
settings = "--prune '0 0 1' -T $working-dir/lm -S 20%"
order = 5
lm-binarizer = $moses-bin-dir/build_binary
type = 8

[LM:kw-tc-wn]
raw-corpus = $bitext-dir/train.$output-extension

[TRAINING]
script = $moses-script-dir/training/train-model.perl
training-options = "-mgiza -mgiza-cpus 4 -sort-buffer-size 8G
                    -sort-compress gzip"
parallel = yes
alignment-symmetrization-method = grow-diag-final-and
lexicalized-reordering = msd-bidirectional-fe
max-phrase-length = 5
score-settings = "--GoodTuring --MinScore 2:0.0001"

[TUNING]
tuning-script = $moses-script-dir/training/mert-moses.pl
tuning-settings = "-mertdir $moses-bin-dir"
raw-input = $bitext-dir/dev.$input-extension
raw-reference = $bitext-dir/dev.$output-extension
nbest = 100
filter-settings = ""
decoder-settings = "-threads $cores"

[TRUECASER]
trainer = $moses-script-dir/recaser/train-truecaser.perl

[EVALUATION]
decoder-settings = "-search-algorithm 1 -cube-pruning-pop-limit 5000
                    -s 5000 -threads $cores"
multi-bleu = "$moses-script-dir/generic/multi-bleu.perl -lc"
multi-bleu-c = $moses-script-dir/generic/multi-bleu.perl
analysis = $moses-script-dir/ems/support/analysis.perl
analyze-coverage = yes
report-segmentation = yes
```

```
[EVALUATION:kw-tc-wn]
raw-input = $bitext-dir/test.$input-extension
raw-reference = $bitext-dir/test.$output-extension

[REPORTING]
# no parameters to set
```

A.2 ERG

The 1214 version of the English Resource Grammar serves as the base version I use for my experiments. While a compiled binary image of this grammar is available,[3] there are a few modifications I make for it to work better with transferred semantics as the input for generation. I therefore use the source version, which can be obtained via Subversion:

```
~$ svn co http://svn.delph-in.net/erg/tags/1214 erg-1214
~$ cd erg-1214/
```

In the above session, I also changed the working directory to the freshly downloaded erg-1214/ directory to assist in the following commands.

In this version of the ERG, the default semantic hierarchy (defined in the SEM-I) for quantifiers is not sufficiently distinguished for transfer from Japanese. Specifically, the abstract quantifier type does not distinguish articles from demonstratives, which leads to undesired realizations. I therefore add some additional elements to the hierarchy so these can be properly distinguished. I do this by creating a new file (etc/jaen.smi) for the new definitions, and by including the new file in the top-level SEM-I file.

```
~/erg-1214$ cat <<EOF > etc/jaen.smi
predicates:
  def_udef_a_q < existential_q.
  def_explicit_q < def_udef_a_q.
  def_implicit_q < def_udef_a_q.
  udef_q < def_udef_a_q.
  _the_q < def_udef_a_q.
  _a_q < def_udef_a_q.
EOF
```

[3] http://sweaglesw.org/linguistics/ace/download/erg-1214-x86-64-0.9.26.dat.bz2

```
~/erg-1214$ echo "include: jaen.smi" >> etc/erg.smi
```

There also exist lexical entries that may be output by the transfer grammar that are not added to the ERG lexicon by default. These can be extracted with the `mtr-to-lexicon` script of the XMT software. The resulting lexicon (`jaen-lexicon.tdl` below) must be added to the grammar script, as well.

```
~/erg-1214$ ${XMTROOT}/scripts/mtr-to-lexicon.py --sem-i etc/erg.smi \
                typemap.json ${JAENROOT}/jaen/single-selected.mtr \
                > jaen-lexicon.tdl
~/erg-1214$ cat <<EOF >> english.tdl

;; Lexicon augmentations for JaEn transfer outputs.
:begin :instance :status lex-entry.
:include "jaen-lexicon".
:end :instance.
EOF
```

A bug exists in the `1214` version of the ERG where some constructions result in an argument of `*top*` instead of a valid variable. This bug is fixed in the trunk branch of the ERG, but not in the version I use. To fix it, replace `[ ARG1 *top* ]` with `[ ARG1 semarg ]` in the definition of `basic_arg01_relation`.

Appendix B

DATABASE SCHEMA

Here I describe the tables used to process my translation pipeline as described in Chapter 4. There are four stages: input items, parsing, transfer, and generation. There is only one table for items, but each of parsing, transfer, and generation have two tables: one with time and memory information for each item (where such data is available from the processor), and another with the results for each item, if any.

field	description
`i-id`	item id
`i-input`	source sentence
`i-length`	number of tokens in the source sentence
`i-translation`	target (reference) sentence

Figure B.1: `item` Table

field	description
`i-id`	item id
`time`	processing time (msec)
`memory`	bytes of memory allocated

Figure B.2: `p-info` Table

field	description
`i-id`	item id
`p-id`	parse id
`mrs`	source semantic representation
`score`	parse reranker score

Figure B.3: `p-result` Table

field	description
`i-id`	item id
`p-id`	parse id
`time`	processing time (msec)
`memory`	bytes of memory allocated

Figure B.4: `x-info` Table

field	description
`i-id`	item id
`p-id`	parse id
`x-id`	transfer id
`mrs`	target semantic representation
`score`	transfer reranker score

Figure B.5: `x-result` Table

field	description
i-id	item id
p-id	parse id
x-id	transfer id
time	processing time (msec)
memory	bytes of memory allocated

Figure B.6: g-info Table

field	description
i-id	item id
p-id	parse id
x-id	transfer id
g-id	realization id
surface	target sentence
mrs	fully specified target semantic representation
score	parse reranker score

Figure B.7: g-result Table