Enhancing Machine Translation for English-Japanese Using Syntactic Pattern Recognition Methods

By
Thomas McMahon

A thesis submitted to
the Faculty of Graduate and Postdoctoral Affairs
in partial fulfilment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

January 2015

© Copyright
2015, Thomas McMahon

The undersigned hereby recommend to
the Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis,

Enhancing Machine Translation for English-Japanese
Using Syntactic Pattern Recognition

Methods

submitted by

Thomas McMahon

Dr. Douglas Howe
(Director, School of Computer Science)

Dr. B. John Oommen
(Thesis Supervisor)

Carleton University
January 2015

ABSTRACT

In this thesis, we present a novel approach to machine translation using syntactic Pattern Recognition (PR) methods. The purpose of this research is to evaluate the possibility of using syntactic PR techniques in this field, as well as to identify any potential benefits in such an approach. To make use of syntactic PR techniques, we propose a system that performs string-matching to pair English sentence *structures* to Japanese *structures*, with the goal of facilitating translation between the languages. In order to process the sentence structures of either language as a string, we have created a representation that replaces the tokens of a sentence with their respective Part-of-Speech tags, using a hybrid tag set created for this system. To perform the actual string-matching operation we make use of the OptPR algorithm, a syntactic award-winning PR scheme that has been proven to achieve optimal accuracy. Through our experiments, we show that our implementation obtains superior results to that of a standard statistical machine translation system on our data set, with the additional guarantee of generating a known sentence structure in the target language. With further research, this system could be expanded to have a more complete coverage of the languages worked with, given the capability to handle more complex sentence structures.

ACKNOWLEDGEMENTS

This thesis is the cumulative result of the time and effort I have spent as a graduate student here at Carleton University. Whether directly or indirectly, this final product has been affected by many people. With their influence this thesis has reached its completed state, and so I would like to thank them here.

First, I would like to express my gratitude towards my parents for supporting me in my academic endeavours, both financially and spiritually, and trusting in the various decisions I have made in that time.

Secondly, I would like to thank my supervisor, Professor B. John Oommen, in both providing the initial inspiration for this thesis, as well as patiently working with me in producing this final document. Without his aid in preparing this thesis, I am confident that it would not nearly be of the same level of quality.

I would also like to thank Nathan Bell, a fellow student who was gracious enough to spend time revising sections of this document, and for providing me with several suggestions that have improved the overall presentation of this work.

Additionally, I would like to thank Professors Diana Inkpen and Jean-Pierre Coriveau for providing important feedback that has helped finalize this thesis.

Finally, I would like to thank all of my friends who, though they may not know it, have provided me support and inspiration through observing their own endeavours this past year.

Contents

1	Introduction	2
1.1	Overview	2
1.2	Motivation of the Thesis	3
1.3	Objective of the Thesis	5
1.4	Contributions of the Thesis	5
1.4.1	Goals and Scope	6
1.4.2	Evaluation of Results	7
1.5	Outline of the Thesis	7
1.5.1	Survey of the Field (Machine Translation)	7
1.5.2	Survey of the Field (Pattern Recognition)	7
1.5.3	Contributions (Tagging Aspects)	8
1.5.4	Contributions (Syntactic PR)	8
1.5.5	Experimental Setup and Results	8
2	Survey of the Field (Machine Translation)	9
2.1	Machine Translation	9
2.1.1	Challenges	10
2.1.2	Systems of Translation	13
2.1.3	Natural Language Processing (NLP) Techniques	29
2.2	English-Japanese Machine Translation	35
2.2.1	Challenges of English-Japanese Translation	35
2.2.2	Translation by Analogy	37
2.2.3	Example Based Machine Translation	41

2.2.4	Syntax-Based Statistical Model for Machine Translation . . .	48
3	Survey of the Field (Syntactic PR)	51
3.1	Introduction	51
3.2	Methods of Pattern Recognition	51
3.2.1	Pattern Recognition	51
3.2.2	Statistical PR	52
3.2.3	Syntactic Pattern Recognition	53
3.3	Syntactic PR: A Hierarchical Approach	53
3.3.1	Constructing Representations using Formal Language Theory	54
3.3.2	Classification using Hierarchical Syntactic PR	55
3.4	Syntactic PR Applications in NLP	56
3.4.1	Overview of the Structural Semantic Interconnection Algorithm	57
3.4.2	The SSI Algorithm	58
3.4.3	Experimental Results	60
3.5	Information Theoretic Syntactic PR	62
3.5.1	The OptPR Model	62
3.5.2	The OptPR Algorithm	65
3.5.3	Experimental Results	67
3.6	Conclusion	68
4	Contributions: Tagging Aspects	69
4.1	Introduction and Motivation	69
4.1.1	Chapter Overview	69
4.1.2	Context	70
4.1.3	Purpose	70
4.1.4	Rationale	70
4.1.5	Roadmap	71
4.2	Contributions of this Chapter	71
4.2.1	Contribution: Template Sentences	71
4.2.2	Contribution: Tagging Methods	72
4.2.3	Contribution: Tag Simplification	72

4.2.4	Contribution: Phrase-Level Template Parsing	73
4.3	Template Sentences	73
4.3.1	Purpose of Contribution	74
4.3.2	Description of Contribution	74
4.4	Tagging Methods	78
4.4.1	Purpose of Contribution	78
4.4.2	Description of Contribution	79
4.5	Tag Simplification	82
4.5.1	Purpose of Contribution	82
4.5.2	Description of Contribution	82
4.6	Phrase-Level Template Parsing	84
4.6.1	Purpose of Contribution	84
4.6.2	Description of Contribution	85
5	Contributions: Enhancements Using Syntactic PR	87
5.1	Introduction and Motivation	87
5.1.1	Chapter Overview	87
5.1.2	Purpose	88
5.1.3	Roadmap	88
5.2	Contributions of this Chapter	89
5.2.1	Contribution: Word Alignment Utilization	89
5.2.2	Contribution: Sub-Sentence and Language Division	89
5.2.3	Contribution: MT Using OptPR Algorithm	91
5.3	Word Alignment Utilization	92
5.3.1	Purpose of Contribution	92
5.3.2	Description of Contribution	92
5.4	Sub-Language Models	96
5.4.1	Purpose of Contribution	97
5.4.2	Description of Contribution	97
5.5	MT Using OptPR Algorithm	98
5.5.1	Purpose of Contribution	99

5.5.2	Description of Contribution	99
5.6	Conclusion	101
6	Experimental Setup and Results	102
6.1	Introduction	102
6.1.1	Chapter Overview	102
6.1.2	Road Map	102
6.2	Data Acquisition and Processing	103
6.2.1	Grammar Used in Experimentation	104
6.3	Experimental Configurations	105
6.3.1	Setup-1: System with Primary Data Set	106
6.3.2	Setup-2: System with Syntactic-Filtered Data Set	106
6.3.3	Setup-3: System with Tag Simplification	106
6.3.4	Phase-1: Base String Matching	107
6.3.5	Phase-2: Content Agreement	107
6.3.6	Phase-3: Dictionary Matching	108
6.4	Baseline Experiment	108
6.4.1	Moses Statistical Machine Translation System	108
6.4.2	IRST Language Modeling (IRSTLM)	109
6.4.3	GIZA++ Word Alignment Tool	109
6.4.4	Baseline Results	109
6.5	Experimental Results and Analysis	110
6.5.1	Setup-1	110
6.5.2	Setup-2	112
6.5.3	Setup-3	114
6.5.4	Baseline Comparison	115
6.5.5	Overall Evaluation	118
6.5.6	Standard Evaluation Metrics	119
6.6	Conclusion	121
7	Conclusion and Future Work	122
7.1	Introduction	122

7.1.1	Required Data Size Reduction	122
7.1.2	Syntactic Translation Performance	123
7.1.3	Syntactic Accuracy of Translation	123
7.2	Contributions	124
7.2.1	Contribution 1 in Chapter 4: String Representation of Sentence Structures	124
7.2.2	Contribution 2 in Chapter 4: Phrase-Level Template Parsing .	125
7.2.3	Contribution 1 in Chapter 5: Word Alignment Utilization . .	125
7.2.4	Contribution 2 in Chapter 5: Machine Translation using the OptPR Algorithm	126
7.3	Conclusion	127
7.4	Future Work	127
	Bibliography	129

List of Figures

2.1	Content words provide the meaning of the sentence, while function words communicate the syntactic relationships between them.	10
2.2	The idiom “He kicked the bucket” literally translates into Japanese as someone kicking a bucket, but is intended to mean “He died”.	12
2.3	Mapping of the English sentence “We went to the hospital” to its Japanese translation. The sequence of the components of the sentence is not the same in the two languages. The Japanese transliteration is meaningless.	12
2.4	The Vauquois Triangle that shows the effort of analysis and synthesis, as it relates to closing the “distance” between two languages.	13
2.5	This figure shows the basic idea behind a Transfer-Based translation system. The three-stage process can be used to understand other translation methods as well.	14
2.6	A constituent parse, showing the syntactic structure of the sentence “We went to the hospital”.	18
2.7	Resulting constituent parse tree for the tree given in Figure 2.6 after the Transfer phase of translation. The structure is transformed to fit Japanese syntax, and mapped to the respective Japanese words. . . .	19
2.8	Output Japanese sentence for the sentence “We went to the hospital” using the Syntactic Transfer.	20
2.9	After identifying the semantic roles of the elements in the source text, the same roles can be preserved during transfer, resulting in only the words needing translation.	20

2.10	A simplified ontology of English nouns. Each child node in the tree is a more specific instance of the parent node. The dotted line represents a jump ahead in the ontology.	23
2.11	Interlingual representation focuses on capturing the semantic data of the source text. For the purpose of this demonstration, the ontology used has an English representation.	24
2.12	A potential word alignment for the example sentence. Substitutions, deletions, and insertions are all needed to construct the translation model, and yet the translation model is needed to approximate the word alignment. This reliance results in the need for an iterative process.	27
2.13	Assuming that the correct sense of the verb is known, the arguments contained in the sentence can be assigned to certain slots required by the verb. If a potential sense of a noun does not fit the required type of argument, it can be rejected in favour of a sense that does fit the verb.	33
2.14	Certain verbs use the particle “ga” to mark the object. Other sentences use “ga” instead of the usual subject marker “wa” to emphasize that the subject is the topic of the sentence.	37
2.15	The original English sentence is shown on left, and the romanized Japanese equivalent on right side. By changing the destination of the sentence, the translation changes slightly, showing the position of the destination in the Japanese sentence.	38
2.16	A simple example of translation by analogy. From the case frame representation, the similarity between the pre-translated sentence and the new one can be used to create the output.	40
2.17	Constituent parse (left) and Dependency parse (right) of the sentence “We went to the hospital”. In a dependency parse, each node consists of a word, meant to show the syntactic dependencies that the words of a sentence have with each other.	43
2.18	This figure shows various translation units. Since the equivalent structures are known for “We went to the hospital” and “to the hospital”, a third translation unit is created, where the “X” marks an insertion point.	44

2.19	By matching the input to pre-existing translations, a patchwork translation can be performed. Matching Expressions (MEs) use a notation that supports the addition, replacement, and deletion of dependency sub-trees in an attempt to form the source text. The ME formed is used to facilitate translation.	45
2.20	Logical Form (LF) for “We went to the hospital”. The LF captures semantic roles for the subject and the object in regards to the verb, along with additional information.	47
3.1	An example parse using a CFG, using a subset of English grammar. Lower case words indicate terminal symbols, while words beginning with Upper case indicate a non-terminal node.	55
3.2	A simplified semantic graph displaying the basic concept of the relational graphs constructed. In reality, each node would branch off with numerous relations, with a maximum distance of three.	58
3.3	The words selected for the context when working with the nouns of the word-sense “glass”. The nouns in blue are from the actual word-sense, and are the words that are used to measure the success of the algorithm. The words in red are from the hypernym “container”, and are not used in measuring the success of the algorithm, but in providing additional context.	61
3.4	The three distributions of the model are used to generate a noisy string Y from U	64
3.5	The language model generates a TL sentence, e , that is transmitted through the channel’s translation model into the SL sentence, f . The original sentence e is then estimated with a decoder function obtaining $\hat{e}$	64
3.6	The algorithm calculates the probability of string Y being derived from string U , based on the distributions Q , G , and S . The algorithm operates in cubic space and time.	66

4.1	Replacing the words of a sentence with their syntactic classes results in a “Template Sentence”, which can be used to form different sentences with the same structure.	75
4.2	Single and Multi-Word translation can be performed using templates, as long as a mapping function exists.	76
4.3	When mapping outside of the content classes, words are often mapped to different classes or simply dropped/added.	77
4.4	The figure shows how a Japanese TS can be acquired using the ChaSen tagger. Due to the syntactic importance of Japanese particles, they are preserved in place of their tags.	80
4.5	This figure shows that by selectively converting ChaSen tags into Penn Treebank tags, a Japanese TS can become comparable to an English TS.	81
4.6	By simplifying the templates in ways that allow information to be reinserted, the performance of the mapping algorithm is potentially increased.	83
4.7	Using phrase-level matching for TSs allows for sentence structures that are not covered by the corpus to be generated.	85
5.1	The t3 file contains a probability distribution for the one-way mapping between the SL and TL, based on the parallel corpus provided. As the OptPR algorithm operates by assuming the input is a noisy version of a dictionary entry, the Japanese language is considered the SL in order to obtain the proper distributions.	94
5.2	The probability distribution obtained from running the word alignment tool with the Japanese language as the SL and the English language as the TL. The non-null Japanese tag rows are used for the substitution distribution, and the null Japanese tag row is used for the insertion distribution.	95
5.3	The probability distribution obtained from running the word alignment tool with the Japanese language as the SL and the English language as the TL. The null English row is used for the deletion distribution.	96

5.4	Using the OptPRs probability score, the input phrase can be ranked against the phrases contained in the algorithm’s dictionary, which can then be used to calculate the most probable sentence structure. . . .	98
6.1	Line Graph of the results for Setup-1. The peaks of each phase can be seen to migrate closer to the left side, indicating an increase in the overall accuracy.	111
6.2	Line Graph of the results for Setup-2.	113
6.3	Line Graph of the results for Setup-3.	114
6.4	Comparison graph between the baseline SMT results and the results of Phase-1. Baseline results are slightly superior, but they are comparable.	116
6.5	Comparison graph between the baseline SMT results and the results of Phase-2. The results of Phase-2 show our scheme to be slightly superior to that of the baseline.	117
6.6	Comparison graph between the baseline SMT results and the results of Phase-3. While the results of Phase-1 show that the OptPR algorithm can produce comparable results to the baseline with the benefit of syntactic correctness, Phase-3 shows that with further processing our system is capable of surpassing the baseline results while still maintaining that correctness.	118

ACRONYMS

Chapter 1

Introduction

1.1 Overview

This thesis is concerned with the field of research known as Machine Translation (MT). This involves the task of a computer translating the text written in one language into that of another without the need of human intervention. Specifically, we are interested in English-Japanese MT, a special case where English text is automatically converted into equivalent text belonging to the Japanese language.

MT is a sub-domain of Natural Language Processing, a field of research concerned with transforming the text of natural languages into a form that can be understood and manipulated by machines. The term “Natural Language” refers to languages that evolved naturally in human society, as opposed to artificially constructed languages, or explicitly-defined languages that lack any evolutionary development. This distinction is important, as natural languages have changed constantly throughout history, and so the way in which sentences are formed can never be completely defined. A natural language contains too many exceptions to be completely covered by any rigid set of rules, as this set of rules would have to constantly adapt to drifts in common usage. This is one of the many difficulties associated with properly handling languages programmatically.

Translation between certain languages is a difficult task even for humans, as the meaning must be correctly interpreted from the original text and properly represented

in the accompanying translation. The meaning conveyed in a sentence is potentially impossible to completely recreate when rewritten in another language for many reasons, one of which is that a natural language is also a reflection of its associated culture. Certain aspects of a language may be more important to one culture than another, and so those portions will have more minute distinctions in meaning. To handle this, some amount of decision-making must be incorporated so as to produce the closest possible translation. Performing MT is an even greater challenge, as faculties naturally available to humans, such as the ability to grasp the *intended* meaning, must be artificially replicated to yield truly accurate translations. English-Japanese translation is a particularly difficult language-pairing, as there is no known common ancestry between the two languages, resulting in fundamentally distinct sentence structures [12].

One approach to handling the translation of two such structurally-distinct languages is to create a method of transforming the structure of one language so that it matches the other. One potential way of accomplishing this is to compare the structure of an English sentence to every possible Japanese sentence structure, and to somehow be able to tell which pairing is the best match. The issue then is one of determining which pairing is ideal. There is a class of algorithms that are capable of matching structures in this way, i.e., using Syntactic Pattern Recognition techniques. Making use of such algorithms could potentially help in improving translation between two languages as distant as English and Japanese. This is the primary focus of this thesis.

1.2 Motivation of the Thesis

There are many issues that are inherent to MT as languages were simply not developed with such a process in mind. Despite this, advances in MT have been made throughout the years to produce increasingly more accurate results, but these techniques are far from perfect. In this thesis we propose and implement a number of potential strategies for dealing with problems preventing ideal translation. The following are some of the issues encountered in this field.

A primary concern in MT is that in order to perform translation between two languages, one must devise a new system for that specific language pairing. In terms of a MT system, a source language is the language of the input text, and the target language is that of the output. Each unique combination of source and target languages requires a new translation system to be constructed. There is no single approach to MT that will perform satisfactorily for every language pair, and so various methods must be constructed and tested in order to obtain a sufficient model.

When performing MT, the end result is a sentence that may or may not actually belong to the target language. For a sentence to belong to a language, it needs to have proper structure and to also be composed of words belonging to that language's lexicon. As mentioned previously, there is no complete representation of a natural language's syntax, and so it is difficult to automatically verify that a sentence belongs to a certain language using a set of rules. A system that does this requires a massive amount of either manually-defined or automatically discovered rules, and neither approach can guarantee the construction of a rule set that can completely cover a language [19]. The alternative would be to have an exhaustive list of all potential sentences of a given language, but, obviously, this is not a feasible approach. A common approach in MT uses statistical methods, to instead construct a "Language Model" that attempts to act as such a resource, by calculating the probability of a given sentence existing in a language [30]. The issue with such an approach is that it is *only* an approximation and so will generate sentences that are not quite grammatically correct.

Depending on the approach, a MT system can potentially be made up of several components. As natural languages were not designed to be understood by machines, each component has a very real chance of introducing inaccuracies in the translation. When several components of a MT system are used together, it follows that the chances of producing an accurate translation would be dramatically reduced.

1.3 Objective of the Thesis

While we do not address every issue listed in the above section, we hope to be able to provide potential solutions for a number of them. The following problems are addressed in this thesis:

- (a) Required Data Size Reduction: We hope to reduce the amount of data required to sufficiently train a MT system.
- (b) Syntactic Translation Performance: By improving the overall accuracy of one component in the MT system, we can both improve the resultant translation and also mitigate the effect of stacking failures across several components of the system.
- (c) Syntactic Accuracy of Translation: Using a syntactic Pattern Recognition (PR) approach, we hope to increase the likelihood that the output of a translation system is syntactically accurate.

1.4 Contributions of the Thesis

In order to achieve the objectives of this thesis, we have opted to take a novel approach by incorporating syntactic PR methods, combined with our own representation for sentence structures. The contributions of this thesis are as follows:

- (a) String Representation of Sentence Structures: We make use of a unique representation of sentences that corresponds to their structure rather than their meaning. These sentence structures are stored instead of an exhaustive list of sentences, to replace the approximations of a language model.
- (b) Phrase-Level Template Parsing: In order to account for the distance between languages, we break down their full sentences into components. Using these, we can train models to handle specific phrase structures, and to also mitigate the issue of word ordering.

- (c) Word Alignment Utilization: We re-purpose existing statistical MT components to aid in the construction of our syntactic PR system. These statistical components are typically used for large corpora of sentences, and so we evaluate the effectiveness of word alignment on smaller data sets using our structural representation of sentences.
- (d) Machine Translation using the OptPR Algorithm: By making use of our string-based representation and word alignment models, we choose to use a syntactic PR algorithm (the OptPR algorithm [28]) to effectively match English structures to Japanese structures contained in a set list, so that accurate translation can be performed with the guarantee that the output will have a known Japanese sentence structure.

1.4.1 Goals and Scope

The goal of this thesis is to show the viability of using syntactic PR methods in the field of MT. The potential benefits of using such an approach are great, particularly in being able to ensure that the output of the translation system is, at least, structurally sound.

Before we proceed, it is pertinent for us to highlight the scope of the thesis. This thesis does not present a complete MT system – embarking on the latter would be the task of a lifetime. Rather, our goal is to demonstrate the applicability of syntactic PR tools, that have not been previously used in MT. In that sense, we are proposing the use of these tools in MT components that could potentially be *used* in more comprehensive MT systems. Further, since our aim is to demonstrate a *prima facie* case for these tools, as it stands, our component essentially takes as its input an English sentence with a rather simple form, i.e., it contains a Subject Phrase, a Verb and an Object Phrase, and these are labelled with Part-of-Speech (POS) tags. The module matches it against a collection of potential Japanese sentence structures that also contain analogous POS tags.

Although this current implementation is limited to simplistic sentence structures, we believe that in the future, with some effort, it could be expanded upon in order to

increase the complexity of sentences handled, by recursively breaking the larger and more-complex sentences into phrases that the current component can process.

1.4.2 Evaluation of Results

The results obtained through our experiments are promising, showing that by merely using the OptPR algorithm in combination with the word alignment models, we can attain results comparable to that of our baseline statistical MT system, which makes use of more sophisticated models [9]. Our system also comes with the guarantee of producing a sentence structure that is known to exist in the Japanese language. Through further augmentation of our initial system, we are also able to improve the results, outperforming the baseline system.

1.5 Outline of the Thesis

1.5.1 Survey of the Field (Machine Translation)

The field of MT encompasses a large body of research and applications involving many distinct approaches. This chapter will provide an overview of this field by exploring many of the approaches and challenges in the general area of MT and also in the specific sub-domain involving English-Japanese translation. By doing so, we hope to provide the context to understand the difficulties associated with performing any work involving natural languages, as well as in justifying our methods.

1.5.2 Survey of the Field (Pattern Recognition)

A major component of our project involves a syntactic PR algorithm, and so this chapter focuses on defining PR, and more specifically, the two schools of statistical and syntactic PR. By contrasting syntactic PR with statistical methods, we can both define what a syntactic PR approach is, as well as why it is more appropriate for our problem. Thereafter, we explore various forms of syntactic PR, showing how the OptPR algorithm, an award-winning result, is best suited for our problem domain.

1.5.3 Contributions (Tagging Aspects)

In order to utilize a string-matching algorithm, we undertake a novel approach described in terms of using tags for MT. In this chapter, we will survey previous research that has used similar approaches in MT so as to display the past successes and to also inspire our approach. Subsequently, we will explain how the paradigm of “Template Sentences” used in our experiments are obtained and utilized effectively.

1.5.4 Contributions (Syntactic PR)

Statistical approaches are commonly used in MT, and complete translation systems have been built using purely statistical models. Since we are using a syntactic approach, this chapter will show how we can utilize previously-existing tools that are commonly used in statistical MT for our approach. We will then show how these models are adapted and used in combination with the OptPR algorithm to perform the string matching between English and Japanese sentences.

1.5.5 Experimental Setup and Results

This chapter will provide the specifics regarding the experiments conducted, as well as the results obtained to evaluate our system. These results are then analyzed to determine the benefits and drawbacks of using our newly-proposed approach in MT.

Chapter 2

Survey of the Field (Machine Translation)

2.1 Machine Translation

This section will cover machine translation, a field within the domain of Natural Language Processing (NLP). First, we formalize the concept of machine translation and follow this by describing the various challenges inherent to the task. Afterwards, we cover various approaches that have been created to deal with these issues.

Machine Translation is the process by which a machine automatically converts text from one natural language into another. The language in which the text originated from is referred to as the “Source Language” (SL), and the language of the output text is called the “Target Language” (TL).

The primary goal of a translation system is to preserve the intended meaning of the source text by preventing any loss of information. There are numerous instances where a TL cannot fully represent the meaning of the source text, and so loss of information is inevitable. Ideally, this situation should be the only instance of information loss within the system, but various challenges associated with working with a natural language cause loss at other junctures.

The secondary goal of a translation system is to correctly translate the input text into the TL, with no *grammatical* errors. Depending on the differences between the

source and target languages, this too can be a difficult task.

These challenges are discussed in the following section.

2.1.1 Challenges

There are a variety of challenges that arise whenever a natural language is involved, many of which are consequences of the various ambiguities contained in any text. Ambiguities arise whenever a decision must be made, and they can be introduced at the syntactic, semantic, and pragmatic levels of natural languages (each of which are covered below). Ambiguity resolution is especially important in Machine Translation, because all ambiguities must be resolved before a correct translation can be achieved.

A problem that is more specific to Machine Translation is the differences in the grammars of the two languages in question. Typically, closely-related (those from a similar “ancestry”) languages will have similar grammars, while languages with distant or no shared ancestry may have grammars that are completely distinct from each other. The consequences of this are covered later on.

Syntactic Ambiguity

Syntactic ambiguity occurs when a word in a language can potentially belong to more than one syntactic word-class. Using English as an example, words can be separated into two major classes: “function” words and “content” words. Function words contribute to the syntactic *structure* of the sentence, whereas content words contribute to its meaning. The image below shows the distinction between the two classes.

Figure 2.1: Content words provide the meaning of the sentence, while function words communicate the syntactic relationships between them.

This distinction between words is important as, typically, content words are more likely to be preserved in a translation than function words. Function words communicate the syntax of a language, and there may be no equivalent syntax in the TL. What is important in Machine Translation is the task of capturing the relationships between the content words shown in the syntax, not necessarily the syntax *itself*.

Syntactic ambiguity occurs more heavily within the subclasses of content words. These sub-classes are: noun, verb, adverb, and adjective. In English, almost every verb can be used as a noun. Additionally, the standard suffix for the present-tense verb form is the same as the standard suffix for indicating a plural noun. Many other ambiguities exist in both the content and function word-classes, including words that can be both a content and function word (e.g. “there” can be a pronoun (function) or an adverb (content)).

Semantic Ambiguity

Semantic ambiguity occurs when a word refers to two separate meanings. For example, the word “bass” can refer to either a type of fish or to a frequency of sound. The corresponding context usually helps in making this distinction, but in other cases it may be vague even to a human reader. Resolving this ambiguity is especially important for Machine Translation, as there is very little chance that the two *potential* meanings will have the same word form in the TL. This ambiguity only occurs for content words, as they are the only words that refer to a meaning.

Pragmatic Ambiguity

Pragmatics in linguistics refers to the “intended” meaning of language usage, as opposed to the “literal” meaning. There are times when the meaning of a sentence has little or nothing to do with the meaning of the individual words it is composed of. Idiomatic phrases are instances of pragmatic ambiguity, as shown below.

Figure 2.2: The idiom “He kicked the bucket” literally translates into Japanese as someone kicking a bucket, but is intended to mean “He died”.

For a translation system to be complete, it would need to deal with situations like these. However, most translation systems have little to no support for pragmatic analysis.

Word Ordering

A simple way of performing translation is to replace each word in the source text with its equivalent in the TL. Assuming perfect disambiguation, this still will almost always result in a nonsensical sentence, because the *order* in which the words occur in a grammatical sentence will differ between languages. This can be seen in the image below, which shows how a simple English sentence maps to its Japanese equivalent.

Figure 2.3: Mapping of the English sentence “We went to the hospital” to its Japanese translation. The sequence of the components of the sentence is not the same in the two languages. The Japanese transliteration is meaningless.

An understanding of the TL’s grammar is necessary for producing quality translations. Languages of similar origin will usually have similar structures that characterize

their grammar, whereas distant languages may have fundamentally different methods for constructing sentences.

2.1.2 Systems of Translation

Originally, the field of Machine Translation was primarily focused on encoding human-designed translation rules to automatically apply a translator's process. As the availability of data (as well as the ability to process it) became more available over time, methods of automatically learning these rules became available, and more complex levels of linguistic analysis became feasible. The Vauquois triangle (shown below) is a diagram that shows this progression of analysis in Machine Translation. It essentially communicates that the “distance” between two languages becomes reduced with higher levels of linguistic analysis, at the cost of system complexity.

Figure 2.4: The Vauquois Triangle that shows the effort of analysis and synthesis, as it relates to closing the “distance” between two languages.

From this triangle, three methods of translation came to be defined. Direct translation is the lowest form, which relies primarily on a bilingual dictionary to translate word-by-word, with little or no analysis. The two middle arrows are referred to as Transfer-Based translation systems. Enough linguistic analysis is done at this stage to create a formalized version of the source text, which can then be used to translate the text into a formalized version of the TL, which, in turn, is synthesized into the output text. Interlingual systems require the most analysis[5]. The source text is stripped away of any dependence on the SL and translated into an “Interlingua”, a language constructed to act as an intermediate between the source and target languages.

These systems are grouped together and referred to as “Rule-Based Systems” (RBS). What defines an RBS is its usage of hand-coded rules and/or its reliance on linguistic analysis to perform translation. The reason for creating this grouping is to distinguish these systems from newer systems of translation, known as “Statistical Machine Translation Systems” (SMT). SMTs treat translation as a statistical problem rather than a linguistic one, and so forgo any and all linguistic analysis in favour of statistical models [35].

Though only a single specific method is referred to as being “Transfer-Based”, all translation systems follow a similar flow of operations to the one shown in the figure below.

Figure 2.5: This figure shows the basic idea behind a Transfer-Based translation system. The three-stage process can be used to understand other translation methods as well.

Though this will be covered more thoroughly in a later section, this diagram shows the three stages for Transfer-Based translation. Though the lower stages are specific to Transfer methods, the three-stage process can be seen as occurring in most translation systems using the upper stage names.

The following sections will cover these four systems of translation, moving up the Vauquois triangle, after which we move on to briefly cover statistical systems.

Direct Machine Translation

Of the three approaches to Rule-Based Machine Translation shown above, a Direct Translation System (DTS) is the simplest. The core of translation in a DTS relies on a bilingual dictionary between the two languages, and little or no preprocessing is performed before the translation phase. The translation phase looks up each individual word in the source text, and replaces it with a word in the TL. One can see that this method promptly encounters two major issues associated with MT: word ambiguity and word order. A single word can potentially map to several words in the TL, due to both class and sense ambiguity. It is thus clear that a decision process is needed to decide which word in the TL is correct. Since the translation happens word-by-word, if there is no re-order operation there is no possibility that the system will produce grammatically correct sentences on any real data. One will certainly need various re-ordering operations primarily because the words are *individually* (and non-contextually) translated. These issues were addressed in the first implementation of a DTS, which was also the first reported MT system.

The first DTS was implemented and demonstrated in 1954, and it was referred to as the “Georgetown-IBM Demonstration” [15]. The purpose of this system was to demonstrate the potential capabilities of automatic translation as achieved by a computer. The SL used was Russian (in its Romanized representation) and the TL was English. Being a DTS, the primary component of the system was the mapping of Russian words to English words. As this system was primarily meant as a demonstration, the dictionary of Russian words was limited to 230 entries, and the English words used were limited to only those that were to appear in the 60 demonstration sentences. This reduced the problem of word ambiguity to a choice between one or

two words. The table below shows example entries in the bilingual dictionary.

Table 2.1: The table shows two examples of dictionary entries of the Georgetown Demonstration. Any Russian word can be mapped to up to two English words/phrases, with the CDD being used as a reference to a rule that determines which word is to be used.

Russian Word	English Equivalent I	English Equivalent II	CDD-1	CDD-2	PID
doma	at home	houses	241	–	151
kachyestvo	quality	the quality	222	–	151

Observe that a single Russian word can have up to two English entries. English entries are allowed to be more than a single word, in an attempt to compensate for the differing syntax between Russian and English. The CDD-1 code determines which English equivalent should be chosen for the translation. The CDD-2 code is used to determine how the word needs to be re-arranged within the sentence so as to obtain a grammatically correct English sentence.

The most important segment of code specified is the PID, which is what determines how the system behaves in regards to CDD-1 and CDD-2. The system translates the input sentence word-by-word, looking at the PID code of the current word in order to determine which English equivalent entry should replace it, and if a certain word in the surrounding context should be re-arranged. There are six possible values for PID codes, listed below along with their system behaviour.

These rules are not founded on any linguistic analysis, but instead, were created to handle very specific sentence structures that are repeatedly used in the 60 sentences used for the demonstration. Because these rules were designed with the demonstration sentences in mind, they do create fluent translations for the demonstration, but would not work with an expanded vocabulary or with sentences that do not conform to the specific structure needed.

Upon analyzing this experiment, it is clear that the achievement of well-translated Russian-to-English sentences is largely due to the usage of pre-prepared sentences in a heavily-reduced scope of both the natural languages. The experiment does more to

Table 2.2: Rules defined to allow system to handle word ambiguity and differing sentence structure between Russian and English.

PID	Rule	Description
110	Rearrangement	If CDD-2 code of previous word ends in “21”, move previous word to be after current word. English Equivalent I is chosen for current word.
121	Choice-Following Text	If CDD-1 code of next word is “221” select English Equivalent I for current word. If CDD-1 code of next word is “222”, select English Equivalent II for current word.
131	Choice Rearrangement	If CDD-2 code of previous word is “23”, retain word order and choose English Equivalent II for current word. Otherwise, choose English Equivalent I for current word, and move previous word to be after current word.
141	Choice-Previous Text	If CDD-1 code of previous word is “241” , choose English Equivalent I for current word. If CDD-1 code of previous word is “242”, choose English Equivalent II for current word.
***	Subdivision	Choose English Equivalent I for current word.

highlight the problems associated with MT rather than with presenting a solution. Specifically, the problems of word-to-word ambiguity and differences in word ordering were the main concerns of the rules defined. The limited number of rules designed to handle these problems sufficed for these 60 sentences, but would not perform nearly as well if the dictionary were expanded for real-world usage. Later research intended to expand on this system showed that this approach was not suitable for general-purpose translation.

Transfer-Based Machine Translation

Transfer-based Translation Systems (TTS) rely more heavily on linguistic analysis than on direct translation systems. As mentioned previously, the basis of a Transfer-Based system is on its three-stage process consisting of analysis, transfer, and synthesis. In the case of a DTS, a bilingual dictionary is used to translate individual words between the two languages. Unlike a DTS, however, a TTS relies more on creating a processable representation of the source text, which can then be transferred into a representation that is more in line with the TL. The Vauquois triangle mentions two levels of transfer: syntactic and semantic transfer.

Syntactic transfer requires that a parse is acquired of the source text during the analysis phase. For the sentence “We went to the hospital” (used previously in an example), the following parse could be made.

Figure 2.6: A constituent parse, showing the syntactic structure of the sentence “We went to the hospital”.

This parse tree is then passed from the analysis phase to the transfer phase, which attempts to transform the syntactic representation into a structure that fits the TL’s

grammar.

With an understanding of the differences between the syntactic structures of English and Japanese, one can apply the following transformations:

Figure 2.7: Resulting constituent parse tree for the tree given in Figure 2.6 after the Transfer phase of translation. The structure is transformed to fit Japanese syntax, and mapped to the respective Japanese words.

Firstly, the verb component (VBD) of the verb phrase (VP) is moved to the end of its scope, to satisfy the Subject-Object-Verb structure of Japanese sentences. Secondly, since Japanese uses postpositions rather than prepositions, “to” is moved to the end of the scope of the preposition phrase (PP) scope. Finally, the specific words are mapped to their Japanese counterpart equivalents. Since Japanese has no determiners, the word “the” maps to nothing.

The final stage in the transfer involves “Synthesis”, where the output sentence is

constructed using the parse tree obtained during the Transfer stage. Since “We” is the subjective form of the personal plural pronoun, the subject marker is added as a postposition to the corresponding Japanese word, resulting in the output sentence:

Watashitachi wa byoin ni ikimashita.
Added as the subject marker.

Figure 2.8: Output Japanese sentence for the sentence “We went to the hospital” using the Syntactic Transfer.

Semantic Transfer systems use this same flow of operations, except that the form used for Transfer is a semantic representation completely abstracted from the syntax. The following image demonstrates a simplistic example of how semantic transfer could be applied to the previous example sentence.

Figure 2.9: After identifying the semantic roles of the elements in the source text, the same roles can be preserved during transfer, resulting in only the words needing translation.

At the cost of more sophisticated analysis, the transfer stage becomes simplified to the point of only requiring word-level translation. The synthesis stage then constructs the output sentence, which also requires more effort.

Interlingual Machine Translation

Interlingual Translation Systems (ITS) [4] perform translation in a similar way to Transfer systems, except that the form obtained in the analysis phase is a completely different language. The basic idea behind an ITS is to translate the SL into an intermediate language (IL), which is then translated into the TL. If another natural language is used as the intermediate language, it is referred to as a “Pivot” language. If an artificial language is used, one that is created specifically for the purpose of performing the translation, it is called an “Interlingua”.

The reason for wanting to use an Interlingua is primarily for allowing translation between multiple languages without the need to develop a new system for each language pair. In a TTS, specific transfer rules are needed for each pair in order to transform the analysis form into a form that can be synthesized into the output. This results in $\binom{n}{2}$ possible language combinations that must be developed. For a system with 10 languages, the number of language mappings required is 45. If an Interlingua is used instead, each language in the system needs to be only used to translate into and out of the Interlingua. The total number of mappings required for the system is then reduced to $(n - 1)$.

The drawback to this approach is that there are two translation steps where information can be lost. The reason for using an Interlingua rather than another natural language is to mitigate this loss, by allowing the Interlingua to be capable of explicitly representing any meaning extracted from the source text. If an Interlingua is able to achieve this, any information loss would only occur during the analysis and synthesis stages of the translation. This means that, ideally, an ITS performs at least as well as a TTS.

However, Interlingual translation requires deeper semantic analysis, as any ambiguity needs to be resolved and all relations between words must be explicitly represented. While it would be beneficial for a system to be capable of doing this, it is not realistic. Semantic analysis on a natural language is a difficult task, and for many languages it cannot currently be performed reliably. Due to the difficulty of transforming text into an interlingual equivalent, most ITSs are for demonstration purposes rather than systems ready for real-world input. Also, there are some cases

in natural language where statements are intentionally ambiguous, and attempting to resolve the ambiguity would affect the intended meaning.

The first step in creating an ITS consists of either defining or adopting an Interlingua. An Interlingua is defined as being composed of the following three elements (S, N, L) :

1. S : A collection of symbols, each of which refers to some semantic data about the input.
2. N : A notation with which the symbols of S can be related to each other to represent the whole meaning of the input.
3. L : A lexicon that relates words of a natural language to a set of symbols from S .

The set S can be represented as a hierarchical tree. The root symbol is the most generic representation of the meaning, and each level of the tree adds more specificity, forming an ontology. An example of a basic ontology is shown below, where “entity” is the root.

Figure 2.10: A simplified ontology of English nouns. Each child node in the tree is a more specific instance of the parent node. The dotted line represents a jump ahead in the ontology.

Information associated with any symbol is then propagated downwards to its children (seen above in the italicized text). The notation N is usually done using nested brackets with propositions to relate the child elements. The Lexicon then transforms each element of a sentence into a set of semantic symbols, all related by propositions. Since the goal of an Interlingua is to explicitly state everything that a natural language communicates, much of which is implied, the interlingual representation can quickly become very complex even for a simple sentence. The running example of “We went to the hospital” is shown below in an Interlingual form.

Event	Walked
Agent	[Person - First, Reflexive Number - Plural]
Goal	[Hospital Number - Singular Definiteness - Definitive]
Aspect	Simple
Tense	Past

Figure 2.11: Interlingual representation focuses on capturing the semantic data of the source text. For the purpose of this demonstration, the ontology used has an English representation.

The first ITS to be used for real-world input is the KANT system [6]. Its primary focus of translation is the domain of technical text, which requires translation to a large number of languages. The system architecture allows for any source and target language to be plugged into the translation system, as the KANT Interlingua completely isolates the two natural languages from being reliant on each other. As the KANT system works with subdomains of languages (technical documentation), a formal grammar can be defined for each language in the knowledge base. The input text is then checked against this formal grammar during the analysis phase to verify that the input is correctly formatted. If it is, the text is transformed into the KANT Interlingua. With a formalized grammar, it becomes easier to interpret word ordering and correctly generate the proper word order during the synthesis stage. The main problem then is word ambiguity. Even in the subdomain of technical documentation, certain words will refer to different meanings. In order to prevent the system from having to resolve these ambiguities itself (which is a difficult problem) the system prompts the user who is inputting the source text to explicitly resolve the ambiguity.

The KANT system was evaluated based on the time it takes to manually edit the

output of the system, versus the time it would take to do the same translation manually. The evaluation showed a 2:1 to 4:1 gain in efficiency, meaning that translators experienced between two to four times efficiency when using KANT versus manually translating it. As this system is designed to allow for human interaction in the pre and post stage of translation, it is closer to the phenomenon of “Computer-Aided Translation” rather than to fully automated “Machine Translation”. However, its usage of an Interlingua makes it a novel system for MT research.

Statistical Machine Translation

The preceding methods all approach machine translation as a linguistic problem. Statistical Machine Translation (SMT) instead approaches it as a purely statistical problem, using the noisy channel concept as a way of modelling translation [30]. Given a sentence s which is to be translated from the SL, it is first assumed that s was originally written as a sentence t in the TL. A series of operations were performed that transformed t into s , and the goal of the SMT is to rediscover the original sentence t .

All sentences contained within the TL have a probability of being the original sentence t , and the job of the SMT is to calculate these probabilities and choose the one that is the most likely. The formula for obtaining the most probable sentence t is:

$$\hat{e} = \operatorname{argmax} P(t|s), \quad (2.1)$$

which, using Baye’s rule, can be calculated using the following:

$$P(t), \text{ and} \quad (2.2)$$

$$P(s|t) \quad (2.3)$$

The first model involves the language model described by Eq. (2.2), which specifies the probability that the sentence t could appear in the language TL. The second model involves the translation model given by Eq. (2.3), which specifies the probability that

the sentence s could have been attained from t . By using these two models, the SMT ensures that both the output of the system is likely to belong to the TL, and that the output is likely to contain the meaning of the source text. To attain a functional SMT, these two models must be generated.

The probability of a particular sentence occurring in a language can be calculated using the following formula:

$$P(t) = P(w_1 w_2 \dots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \dots P(w_n|w_1 w_2 \dots w_{n-1}) \quad (2.4)$$

This means that, for the sentence “The man who had walked into the building was not aware of his surroundings”, the probability of “surroundings” being preceded by the rest of the sentence would need to be known. In any training corpus this sentence is extremely unlikely to appear, and so the language model would calculate a probability of 0 for the sentence, even though it is a grammatically correct English sentence. The sparsity of data in language cannot be overcome by the size of the corpus, but instead by obtaining approximations of the probabilities. One such approximation is attained through the usage of N -grams, which are subsequences of sentences of size N .

For a bigram model, the assumption is made that:

$$P(w_n|w_1 w_2 \dots w_{n-1}) \approx P(w_n|w_{n-1}) \quad (2.5)$$

So, using the previous example, only the quantity $P(\text{surroundings}|\text{his})$ would need to be known, which is far more likely to occur in a training corpus.

To attain the language model, the practitioner simply needs to extract the bigrams from a training corpus, and then, for each word pair the following can be calculated:

$$P(w_1|w_2) \approx \frac{|w_2, w_1|}{|w_2|} \quad (2.6)$$

Additional smoothing techniques can be applied to account for bigrams that did not appear in the corpus, but could still be valid for the language.

Next, the practitioner must approximate the translation model. To acquire this, a sentence-aligned parallel corpus is needed. A parallel corpus is a collection of

sentences written in two languages, so that the equivalent sentences for each language are aligned with each other. The $P(s|t)$ translation model distribution can then be approximated as:

$$P(s|t) \approx \frac{|s, t|}{|t|} \quad (2.7)$$

There are two problems with this approximation. First, it is unlikely that the same sentence will appear in the corpus multiple times, implying that this distribution will almost always have a probability value of unity for each entry. Secondly, it is even more unlikely that the system is translating a sentence contained within the parallel corpus, making this information useless.

Instead, $P(s|t)$ is calculated at the word-by-word level, meaning that the probability of a sentence being translated into another sentence consists of the combined probability of each word in sentence s being translated into a word in sentence t , or otherwise being deleted/inserted. To calculate this, the probability of a word in t becoming a word in s is needed. Additionally, the probability that a word will be deleted in t and the probability of a word being inserted into s is needed. To obtain these distributions, the parallel corpus needs to be aligned at the word level, so that it is known which words are substituted, deleted, and inserted.

Figure 2.12: A potential word alignment for the example sentence. Substitutions, deletions, and insertions are all needed to construct the translation model, and yet the translation model is needed to approximate the word alignment. This reliance results in the need for an iterative process.

However, most parallel corpora are aligned only at the sentence level, requiring a way to automatically align the words for each sentence pair.

The Expectation-Maximization algorithm is used for this purpose, based on the assumption that words that co-occur with each other in the target and source sentences for multiple pairs are more likely to be substitutes for each other. IBM has developed five models for obtaining these distributions, called IBM Models 1-5.

The IBM Model 1 is an iterative process, as it needs to calculate the base translation probabilities for a word in the TL becoming a word in the SL, and then calculating the likelihood that a specific word in a target *sentence* has been substituted for a word in the source *sentence*. This can then be used to obtain a better estimation of the base translation probabilities, resulting in potentially several iterations to obtain the most correct model. To start with, the base translation probabilities are given a uniform distribution, calculated as follows:

$$T(w^t|w^s) = \frac{1}{|W^s|} \quad (2.8)$$

where $|W^s|$ is the number of words contained in the SL. The next step is to apply the current model to the parallel corpus. To do so, the following formula is applied to each sentence pair:

$$p(a|t, s) = \prod_{j=1}^{l_t} \frac{T(t_j|s_{a(j)})}{\sum_{i=0}^{l_s} (t_j|s_i)} \quad (2.9)$$

Where a is a potential word alignment, s is a sentence of the SL, and t is a sentence of the TL. The probability of all potential alignments between two sentence pairs is evaluated with the current translation model. The following formula is then used to count the number of times a specific TL word was aligned with a specific SL word, weighted by the probability of the alignment in question:

$$c(w^t|w^s; t, s) = \sum_a p(a|t, s) \sum_{j=1}^{l_t} \delta(w^t, w_j^t) \delta(w^s, w_{a(j)}^s) \quad (2.10)$$

The maximization step then reassigns the translation probabilities using the following formula:

$$T(w^t|w^s; t, s) = \frac{\sum_{(t,s)} c(w^t|w^s; t, s)}{\sum_{w^s} \sum_{t,s} c(w^t|w^s; t, s)} \quad (2.11)$$

After a certain number of iterations, a translation model is obtained using IBM Model 1. Models 2-5 obtain additional distributions that can be used to further increase the accuracy of the translation model.

Once the Translation and Language models have been obtained, all that is needed is a search algorithm, such as the Viterbi algorithm, to find the corresponding TL sentence given the SL sentence.

2.1.3 Natural Language Processing (NLP) Techniques

The previous section covered various methods for translating between languages, which mainly concerned fitting the meaning of the original sentence into the grammar of the TL. This deals with the problem of word ordering, but assumes that some analysis phase will resolve word ambiguity. As mentioned previously, there are two ways by which a word can be ambiguous: by class and by sense. This section covers two techniques used in general NLP systems to handle these two situations respectively. The first technique is Part-of-Speech (PoS) tagging, which assigns a tag to each token in a sentence corresponding to the word's class. The second technique is Word-Sense Disambiguation (WSD), which analyzes the surrounding context of a word to assign it the correct meaning.

Part-of-Speech Tagging

PoS tagging is the process of automatically assigning a tag to each token (a word or symbol) of a given sentence. A tag is similar to a word-type, except that the tags tend to be more specific than simply “noun” or “verb”. The number of tags depends on the tagset used, but for most English NLP systems the tagset of choice is the Penn Treebank tagset. This is due to it being used in annotating the Wall Street Journal corpus.

The Wall Street Journal (WSJ) corpus is a collection of articles annotated from 1987 to 1989, containing over 25 million words. Among other things, it was annotated for PoS tags. Due to its size and availability, the WSJ corpus has become a standard resource for training statistical NLP models. As it uses the Penn Treebank tagset[22], it also constitutes the standard tags used within the NLP community. Compared to other NLP tasks, PoS tagging is very accurate, achieving more than 97% accuracy on the WSJ corpus. Statistical models are commonly used for this task, and one of the most popular ones being the Hidden Markov Model (HMM).

An HMM assumes that the tag of the current word being processed is dependent on only two things:

1. The probability that a certain tag would be assigned to the current word.
2. The probability that the previous tag in the sentence would be followed by a certain tag

Based on the above, the HMM needs two distributions:

1. The state emission probability: The probability that the tag t would be represented as word w , given by:

$$P(w|t) = \frac{|w, t|}{|t|} \quad (2.12)$$

2. The state transition probability: The probability that the tag t would follow the tag s .

$$P(t|s) = \frac{|s, t|}{|s|} \quad (2.13)$$

The most likely sequence of tags can be calculated using the following formula:

$$\hat{t}_1, n = \operatorname{argmax} P(t_1, n | w_1, n) = \prod P(w_i | t_i) P(t_i | t_{i-1}) \quad (2.14)$$

Using the above formula without modifications would require an exponential time to complete, as it explores the probability of every possible sequence of tags for the given sentence. The sequence can be found efficiently using the Viterbi algorithm,

a dynamic programming approach that finds the most likely sequence of tags in $O(T * |S|^2)$ time, with T being the number of possible tags and $|S|$ being the number of tokens in the sentence.

Word-Sense Disambiguation (WSD)

A “sense” is the meaning referred to by a word token. A given word can potentially be paired with multiple senses, and determining the correct pairing is vital in the semantic analysis of the source text. The Princeton WordNet is a large collection of senses in the English Language along with the corresponding referent words[23]. The collection connects senses in various ways so that it can be seen how closely one sense relates to another. To demonstrate the problem of WSD, consider the various entries for the word “glass” in the WordNet shown in the table below.

Table 2.3: The various senses that the word “glass” can refer to. Even with the knowledge that the word is a noun, a decision between these eight senses must be made.

glass (noun):	1. A brittle transparent solid with irregular atomic structure
	2. A container made of glass for holding liquids while drinking
	3. The quantity a glass will hold.
	4. A small refracting telescope.
	5. An amphetamine derivative.
	6. A mirror.
	7. A collective term for glassware.
	8. A window pane.

One must observe that some of these meanings are more obscure than others, and that some are so closely related that the subtle difference in meaning may not be important in interpreting the word in English.

Whereas PoS tagging is a process that computers perform at a much greater level of accuracy than the average human, WSD is the exact opposite. A human reading

the sentence “I drank water from the glass” can intuitively understand the correct meaning of “glass” without even considering the other possible meanings, while a computer sees a decision problem with eight possible answers. Certain algorithms have been developed to solve WSD, most of which rely on analyzing the surrounding context of a word.

The simplest approach to WSD is to simply choose the most frequently-used sense. With a large corpus of annotated word-senses, the system can count the usage of each sense, and then choose the most frequent sense of the choices for each ambiguity. Rather than use this technique in an actual system, researchers use the results of this approach in establishing a baseline to compare actual WSD systems against. Any approach to WSD should perform better than this baseline, which manages a 28% accuracy.

As with many NLP tasks, WSD can be approached from both a linguistic or a statistical approach. In the above sentence, “glass” is understood as a container since “glass” as a window pane is not used to drink water. Using a resource that contains all of the verbs in the English language, and the roles that objects can play for that particular verb, one can see that there is a way by which WSD can be performed. This technique is referred to as “Selectional Restriction-Based Disambiguation”.

During semantic analysis, relationships between a verb in a sentence and its arguments are extracted. Depending on the verb-sense, certain roles are required, and certain roles are invalid for that verb. If a sentence contains an ambiguous noun, the sense that can fill a mandatory role with the verb is chosen over one that isn’t mandatory or invalid. If the verb sense is ambiguous, the sense that properly aligns with its argument senses is chosen. If both the verb sense and an argument’s sense are ambiguous, the sense-pair that is valid is chosen. The image below shows how the proper sense of “glass” would be assigned using this technique.

Figure 2.13: Assuming that the correct sense of the verb is known, the arguments contained in the sentence can be assigned to certain slots required by the verb. If a potential sense of a noun does not fit the required type of argument, it can be rejected in favour of a sense that does fit the verb.

The limitation of this sort of system is that there are times when an argument contains a noun that is not typically put in the context of a certain verb sense. This is done either to communicate an atypical event, such as someone “eating glass”, or when one is speaking metaphorically.

This approach was used in a limited scope, considering only situations where the verb-sense is unambiguous, which reduced the problem to one of determining the sense of the arguments. This approach resulted in a 44% accuracy rate, an improvement over the baseline. This technique used by itself would not be applicable to real input, because it is more likely that the verb *and* the arguments are *both* ambiguous.

The statistical approach to WSD relies on Machine Learning algorithms (specifically the Naive Bayes Classifier) rather than on any form of semantic analysis[17]. Each word is disambiguated individually without considering the senses of the surrounding context, but rather by only relying on the word-forms and PoS tags. This

prevents paradoxical situations where two ambiguous words in a sentence are reliant on each other's correct meaning. Additionally, this also means that the system is not restrictive of unusual word pairings, as the algorithm will simply choose the most likely sense rather than evaluating a rule that would reject the input.

As a machine learning approach is used, the features to be used for training and testing must be selected. Two sets of features are typically used in WSD: the collocation and the co-occurrence feature sets. The collocation features provide the surrounding context of a word, such as the word forms and the tags of the two left and right surrounding tokens.

Co-occurrence features provide information on whether the word in question appeared with certain words that it typically appears with. This is typically at the sentence level, and so if the word “water” typically appears in sentences with “glass”, its presence in the current sentence would be represented in the feature vector.

Co-occurrence and Co-location features are typically both used in the feature vector v . The following formula is used to get the most probable sense for a particular feature set.

$$\hat{s} = \operatorname{argmax} P(s) \prod P(v_j | s) \quad (2.15)$$

where $P(s)$ is the probability of any sense in the training data being the sense currently being considered. In order to accommodate for data sparsity in the corpus used, each feature in the vector is assumed to be independent of each other. This allows for the second portion of the equation to be used to estimate the probability of the feature vector, based on the individual probabilities of each feature given the sense s .

The problem with any approach in Machine Learning is the need for a large amount of data for a meaningful classifier. Since the feature space for this approach is especially large given the number of words in a language, a large corpus is needed that spans a significant coverage of words and contexts representative of the language as whole, and all of it should be annotated with correct word-senses. Sense-annotated corpora do exist, though the issue of whether or not the amount of data available is sufficient is questionable. To overcome this problem, certain boot-strapping methods can be used to allow a classifier trained on a small amount of data to resolve the

senses of a larger set, which can then be used to further train the classifier.

2.2 English-Japanese Machine Translation

This section covers the issues and approaches used specifically for English-Japanese Machine Translation. English-Japanese MT refers to a specific field where the SL is English and the TL is Japanese. As such, while all of the above methods can be applied to this pairing of languages, not all of them have produced promising results. There is a variety of issues that arise from attempting translation between languages that are very distant in terms of syntax, which will be covered in the following section. Afterwards, we describe the progression of English-Japanese MT. A new method of MT is defined, entitled “Example-Based Machine Translation” (EBMT), designed specifically to handle translation between distant languages. The evolution of EBMT systems follows a similar path to the Vauquois triangle, progressing from dictionary-like matching, to full syntactic and semantic transfer systems. Finally, we survey a statistical system that has been adapted for English-Japanese translation.

2.2.1 Challenges of English-Japanese Translation

This section will cover various challenges that one faces when translating between English and Japanese, whether the task is performed by a human or a computer [12].

There are three levels in which translation between two languages can become complicated:

1. A *lexical* complication occurs when translating a single word from the SL into the TL is more than a simple mapping.
2. A *grammatical* complication occurs when the word structure and semantic information of equivalent sentences are represented in different ways.
3. *Pragmatic* complications involve issues in properly expressing the intended meaning of the source text in the TL (for example, “Kick the bucket”).

Lexical Complications

Plurality: Japanese does not always use morphology to indicate that a noun is plural, as English does. Oftentimes, a noun is read as plural given the context, and is written in the plural as the singular form in Japanese. Translation would have been simple if this was always the case; nevertheless, in other cases the marker “-tachi” is appended to a noun or pronoun to indicate plurality. The task of translation involves the decision as to whether or not one should include this marker.

Lack of Articles: The Japanese language does not use definite or indefinite articles. To represent the meaning of these articles, a different wording has to be used, creating a step that is more complex than translating word-to-word.

Prepositions: Prepositions in English are replaced by postpositional markers in Japanese. Different markers are used depending on the type and semantic role of the preceding noun, and so additional analysis is needed to correctly choose the marker.

Future Tense: Japanese verb phrases do not have a future tense form, and so this information must be communicated in some other way. One potential solution would be to include the adverb corresponding to “future” within the translation.

Verb-Object Correlations: Depending on the object, a verb may take on a different form to communicate the same concept. As an example, the verb “kiru” (to wear) takes on the form “haku” if the object in question is a shoe, and the form “kaboru” if the object is a hat. Thus, when translating a verb from English to Japanese, there is an ambiguity that must be resolved by looking at the *object* noun, and adds a specific mapping that includes this information to the correct Japanese verb form.

Grammatical Complications

SOV Sentence Structure: Whereas English has certain sentence structures that are used to communicate what the subject, verb, and object of a sentence are (using, typically, the Subject-Verb-Object structure), Japanese sentences use particles to communicate this relationship. Because of this, Japanese sentences have a free word order, except that the verb must come at the end.

Subject/Object Markers: Due to markers being used to indicate the subject and the

object of a sentence, these must be determined before translation, and the corresponding marker must be chosen. However, an ambiguity arises when choosing the correct marker, as there are several markers that can be used to indicate either the subject or the object of a sentence, and the particle “ga” can be used for both subject and object markings, depending on the context. The image below provides examples of such cases.

Figure 2.14: Certain verbs use the particle “ga” to mark the object. Other sentences use “ga” instead of the usual subject marker “wa” to emphasize that the subject is the topic of the sentence.

A system that is designed to correctly translate into Japanese must take into account these language differences. These, typically, would be handled during the “synthesis” phase of a transfer system. However, methods specific to handling the translation into Japanese have also been looked at.

2.2.2 Translation by Analogy

The main issue in using rule-based methods, and more specifically transfer-based methods, is that the syntactic structure of the two languages is so different that simply re-ordering nodes of an English constituent tree isn’t sufficient in producing grammatical Japanese output.

Due to the shortcomings of the methods available for English-Japanese machine translation, different methods of translation have been investigated. Instead of relying on word-to-word mappings and minor structural rearrangements, a method was needed that could handle a language pair no matter how structurally distinct they

were. Nagao [25] came up with one such technique, which sought to emulate the learning of a natural language by a human.

According to Nagao, a new language is learned by a student through comparison with a known language. However, he asserts that it is not through analyzing the structural syntax of each language that the student learns. Rather, it is done by memorizing a large number of English and Japanese equivalent sentences with no rigid linguistic analysis. Through this memorization, the student can pick up the repeating structures between the two languages, and deduce how to construct sentences in the new language.

For a computer to learn these repeating structures for English-Japanese translation, a large collection of sentences is needed, with both the English and the corresponding Japanese versions. By inspecting two sentences that only differ by a single word in the English version, the way that a specific word influences the Japanese equivalent can be observed. By repeating this exercise for multiple word positions in a sentence, more information about the structural behavior of the two languages can be inferred. The image below shows how minor changes can be used to observe corresponding structures in two different languages. The example changes the destination of the sentence, and the SVO - SOV difference can be seen.

Figure 2.15: The original English sentence is shown on left, and the romanized Japanese equivalent on right side. By changing the destination of the sentence, the translation changes slightly, showing the position of the destination in the Japanese sentence.

Additionally, Nagao proposed that human translation can also be emulated by a machine. He states that translators do not parse a sentence for its structure,

rearrange words, and then translate it into the TL. Instead, they break the sentence down into phrasal units. These phrasal units are then translated individually based on prior translation of similar phrases and then composed into a sentence in the TL. On this basis, to enable a machine to achieve the same task, both the English and Japanese sentences must be split into corresponding phrases. Since constituent-grammar parsing is not suitable for Japanese (due to its free word order), Nagao proposed the use of a case frame grammar instead.

A case frame grammar is essentially a semantic parsing of a sentence, similar to the Interlingua representation shown previously. It identifies the major verbs used in the sentence, and maps each subject and object to the verbs according to their semantic role. If a sentence and its translation were to be broken down into a case frame parse, certain reoccurring sub-structures could be identified and mapped to each other. When translating a sentence, it could also be parsed into a case frame, and patched together using the identified sub-structures. The example below shows how an existing translation in the system could be used to translate a new sentence, given the same structure.

Figure 2.16: A simple example of translation by analogy. From the case frame representation, the similarity between the pre-translated sentence and the new one can be used to create the output.

As described, this entails that some amount of semantic analysis would have to be done, violating the initial idea of relying on sentence-to-sentence comparison rather than linguistic knowledge. A special scheme for obtaining case frame parses is one that mechanically goes through a large collection of sentence translation pairs. By looking at specific verbs and the nouns that they tend to co-occur with, noun groupings can be created automatically, with no sense of their semantic role. For instance, “hospital” and “restaurant” may both occur in the training data in sentences where the main verb is “went”, in the same position. Since they fill the same role for this verb, they are automatically clustered in the same noun grouping, without doing an analysis that determines that they are both destinations.

Nagao’s research was not implemented into a complete system at the time when he proposed it. But it became the first step in the definition of a new method of machine translation known as Example-Based Machine Translation (EBMT).

2.2.3 Example Based Machine Translation

At its core, an EBMT is a system of translation that uses a parallel corpus as its primary translation utility. Beyond this statement, the boundaries of what an EBMT consists of are vague, as many systems borrow techniques typically seen in a Transfer or SMT system [16].

The use of a parallel corpus of example sentences is not sufficient enough to distinguish EBMT from other systems. This is because SMT uses the same concept to build its language model, and an RBMT could use such a corpus to derive the rules. However, EBMT uses the examples themselves as the *primary* knowledge base, as opposed to any data derived from it. In other words, an EBMT system uses the corpus at runtime, whereas an SMT model uses one to build the translation model *prior* to any actual translation.

The most basic version of an EBMT would be a system that takes an input sentence and attempts to locate it in the parallel corpus. If the sentence is found, it can return the corresponding translation, otherwise it fails. Useful EBMTs would extend on this basic framework to work on a sub-sentence level, and to allow words with similar usage to be substituted for each other. Such a concept is seen in Nagao's case frame idea.

Looking at EBMT systems that can utilize the transfer framework, the analysis phase breaks the input sentence down into phrasal units, the transfer phase attempts to map the units to the TL using the parallel corpus, and the synthesis phase assembles the translated fragments into a proper sentence of the TL.

The benefits to using an EBMT are as follows:

- As the output is based on actual example sentence data, the translation is more likely to be fluent in the TL.
- The system is more sensitive to context and is thus less likely to use incorrect verb-noun pairings.
- Such an EBMT has the ability to handle pragmatic meaning and idioms naturally, since the parallel corpus mapping means that the intended meaning is

preserved between the translations.

- The system is less complex to develop and maintain since only new examples are needed to improve the system, as opposed to complex linguistic entries and rules for an RBMT.

After the initial research and founding of EBMT, a system that uses this model for English-Japanese translation was created by Nagao and Sato [32].

EBMT Implementation

After considering the possibility of translation using a parallel corpus, it becomes obvious that the sentence has to be broken down into phrasal units. This has been mentioned previously, but actually implementing this concept is a challenge. Nagao and Sato proposed the usage of a special representation entitled “Matching Expressions” (ME), which involves the parse of a sentence that provides points at which nodes may be substituted.

The first issue is that the raw parallel corpus is of little use since it simply maps two sentences together. Additional information is needed to see which parts of the sentences can be re-used and swapped for others. To do so, each entry in the corpus is transformed to contain the following three elements:

1. A dependency parse tree of the English sentence,
2. A dependency parse tree of the Japanese sentence,
3. A mapping between the nodes of the two trees showing equivalent sub-sentences.

As mentioned previously, a constituent parse tree of a sentence results in a tree that has as its leaves the tokens of the sentence, after which all parent nodes represent the hidden structure of the sentence. A dependency parse is different as it has a 1:1 correspondence between the number of nodes in the tree and the number of tokens in the sentence. A dependency parse uses the main verb of a sentence as the root, and shows how all other words in the sentence are either directly or indirectly dependent

on it. An example of a dependency parse is shown below, in contrast to a constituent parse of the same sentence.

Figure 2.17: Constituent parse (left) and Dependency parse (right) of the sentence “We went to the hospital”. In a dependency parse, each node consists of a word, meant to show the syntactic dependencies that the words of a sentence have with each other.

The use of a parse tree is suitable for this problem as it allows for sub-trees to be replaced with trees of similar structure, allowing for the concept of using multiple example sentences in translation. The choice of a dependency parse over a constituent parse makes sense given that the Japanese language’s lack of strict word ordering makes a constituent parse impossible.

The assumption made from using a dependency parse for translation is that, while the word order of sentence may change drastically during translation, the dependency structure of the sentence should remain intact.

Instead of phrasal units that were initially proposed to be used for EBMT, “translation units” are used to translate the input sentence. A translation unit is the combined set of the following two items:

1. The sub-tree of any node that has an equivalence mapping stored in the database

2. Any sub-tree from the first set, minus any sub-nodes that also have an equivalence mapping.

The following image shows three translation units, the third of which is the result of replacing the occurrence of the second unit within the first with an insertion point.

Figure 2.18: This figure shows various translation units. Since the equivalent structures are known for “We went to the hospital” and “to the hospital”, a third translation unit is created, where the “X” marks an insertion point.

A Matching Expression (ME) is a notation that allows the combination of various example sentences to form the input sentence. An ME consists of various IDs of

translation units to be used in building the input, as well as various operations to replace, add, or remove certain nodes.

Since each ID corresponds to a translation unit, which has a direct mapping to the equivalent Japanese structure, the English matching expression can be transformed into the Japanese equivalent ME. The Japanese dependency tree is constructed by inserting the various nodes associated with the IDs, and performing the same ME operations used to compose the SME. The output of the system is this Japanese dependency tree, which can then be used to form a sentence. The example below shows the basic concept of translation using MEs. The source input is matched to two entries “we went to the hospital” and “outside”. “outside” replaces “to the hospital” so that the source text is formed using data that has already been translated. The source ME is transferred into the target ME, before finally being synthesized into the TL.

Figure 2.19: By matching the input to pre-existing translations, a patchwork translation can be performed. Matching Expressions (MEs) use a notation that supports the addition, replacement, and deletion of dependency sub-trees in an attempt to form the source text. The ME formed is used to facilitate translation.

The system as it is proposed is essentially a hybrid system using both EBMT and Transfer-based techniques. It requires the input to be linguistically processed

to retrieve its dependency tree, and is presented in a language-dependent special representation, the source ME. The transfer process is the mapping of the source ME to the target ME, and finally the synthesis stage builds the target tree.

One of the initial benefits of EBMT was seen as its relatively simple data-driven approach. Improving the system simply meant adding additional sentence pairs to cover a larger scope of the two languages. Using MEs sacrifices this simplicity by requiring the corpus entries to have dependency parses, and to have direct mappings between the nodes of parallel sentences. By making this sacrifice, the goal of utilizing multiple example sentences for translation is made possible. By accomplishing this, a much larger scope of translatable sentences is allowed, but the technique is still reliant on the words that are being translated appearing in the corpus, with similar usage as the input sentence. This means that the parallel corpus would still need a large amount of data, with the additional cost of parsing and mapping each example. This would either require a large amount of manual labor or the usage of automated parsers or phrase alignment algorithms which, having less than perfect accuracy, would compromise the integrity of the database. As this technique is completely reliant on the example sentence data, this could have a major effect on the translation quality of the system.

English-Japanese EBMT Using Abstract Linguistic Representations

While the previous approach was compared to syntactic transfer-based systems, the following section covers an example-based system that supports semantic transfer [2].

This research continues to build on the concept of using parallel corpora in MT, but takes a different approach than the mapping of dependency structures. The assumption that dependency structures will be retained when translating between a pair of languages may not necessarily be true between two *specific* languages as different as English and Japanese. Dependency parses are still based on syntactic-level co-occurrences, and so languages with completely different syntax structures could generate different dependency trees. What is preserved, or should be preserved, is the meaning of the sentence. By applying semantic analysis, rather than obtaining a syntax-oriented representation, a form that connects the components of a sentence

in a meaningful way can be extracted. The extracted form of the sentence is referred to as the “Logical Form” (LF). An example of a sentence in its LF is shown below.

Figure 2.20: Logical Form (LF) for “We went to the hospital”. The LF captures semantic roles for the subject and the object in regards to the verb, along with additional information.

The first stage of analysis involves constructing a constituent parse of the input sentence. A manually constructed rule-based parser is used for this. There are few hand-crafted parsers that have enough rules to cover real world data, but the one used here is the same grammar parser used for Microsoft Word. The Japanese constituent parser is adapted from the same code as the English parser, with Japanese-specific rules and mechanisms to handle word breaking.

The analysis is achieved by obtaining the LF from the parse.

After analysis, the source LF is translated into the target LF. Before this translation can be accomplished, however, the parallel corpus must be prepared to handle LF transfer.

Similar to the previous approaches, the aligned sentences are broken down into fragments that correspond to each other in the two languages. The difference here is that instead of using a dependency parse tree, the LF tree is used. The alignment algorithm searches for sub-trees in each sentence pair that have similar content words and semantic structure, and extracts these node-pairings. If the particular sub-tree occurs frequently in the corpus, it is then inserted into the MindNet, the repository used at runtime for translation [31].

Once the MindNet has been populated, translation can be initiated. The source LF’s nodes are compared to the nodes stored within MindNet, and a best-fit version

of the source LF is formed using MindNet entries. The target LF is then acquired by replacing each node with the equivalent Japanese version.

The synthesis stage consists of transforming the LF into text of the TL. First, a parse tree is constructed using the LF. The parse tree is then fitted to the TL through various language-specific heuristics. Finally, a sentence is generated from the parse tree, which becomes the output of the system.

This system is capable of performing actual translation, and was compared against a commercial MT system. Both systems translated 232 sentences from English to Japanese, and each sentence was evaluated manually. The translations produced by both systems were presented blindly and rated on both a relative and absolute scale. The relative scale scored the best and worst performer of a particular sentence as a +1/-1, respectively. The absolute scale scored each sentence on the following rating system:

1. Unacceptable
2. Possibly Acceptable
3. Acceptable
4. Ideal

The MSR-MT scored a -0.015 against the commercial product, and a 2.25 on the absolute scale, versus the commercial product's 2.32. These results were shown to be statistically insignificant.

Besides EBMT, other translation methods have also been used for English-Japanese, modified to better handle languages with fundamentally different sentence structures.

2.2.4 Syntax-Based Statistical Model for Machine Translation

In 2001, Yamada and Knight [41] proposed a new way of estimating the translation model to be used in an SMT. It was shown that the IBM models are better suited for word matching between languages that are somewhat similar than for completely foreign pairs such as English and Japanese. As the IBM models work with sequences

and the operations of insertion, deletion and substitution, they may not be sufficient in handling translation that requires large displacement of words. This displacement occurs when translating from an SVO to an SOV structure.

Rather than assuming that the Japanese sentence is a noisy version of the English sentence, it instead assumes that it is a noisy version of the English constituent parse. It is assumed that the English parse has three operations performed on each node: insertion, translation, and reordering. These three operations are covered below:

1. The insertion operation stochastically chooses to insert a Japanese word either before or after the node, or do nothing.
2. The translation operation translates each leaf into either a Japanese word or a null symbol with some probability.
3. The reorder operation stochastically changes the order of the child nodes of the current node. This operation is not applicable to leaf nodes.

There are a number of distributions that are needed. The reorder distribution R is the probability that a node's child sequence s is reordered into the sequence s' . The insertion distribution I^1 is the probability of a word being inserted before, after, or not at all given the label of the current node. The second insertion distribution I^2 is the probability that the word w is inserted. Finally, the translation distribution t is the probability that, given the English word w^e , the Japanese word w^j (or null) is the correct translation.

Once these distributions are known, the most likely word-alignment can be obtained by calculating the combined probability of each operation used to transform an English sentence s into the Japanese sentence t . The issue is then to obtain these distributions. Like the IBM Models, this process is also iterative as it uses the four distributions it is estimating to obtain a more accurate translation model, using the Expectation-Maximization algorithm. To train the model, 2,121 sentence pairs were extracted from a Japanese-English dictionary.

This word alignment procedure was compared against the IBM Model 5 by manually inspecting the word alignments of 50 sentence pairs. Each word alignment in

a given sentence was rated either 1.0 (correct), 0.5 (unsure), or 0.0 (incorrect). If a sentence pair has all alignments rated a 1.0, it is considered to be a perfect alignment. The average alignment score of this approach was 0.582, while the IBM Model 5 achieved an average score of 0.431. Additionally, 10 of the 50 sentence pairs using this method were deemed to be perfect alignments, whereas Model 5 was not able to create even a single perfect alignment. Upon inspection, it was found that IBM Model 5 had misalignments spread out through all fifty sentences, while this method tended to have misalignments concentrated in certain sentences.

Chapter 3

Survey of the Field (Syntactic PR)

3.1 Introduction

This chapter will cover the field of Syntactic Pattern Recognition (PR), including the techniques that will be used in this thesis. First, we define the concept of syntactic pattern recognition and how it differs from statistical methods. Thereafter, we will investigate previous research that has used syntactic PR methods in the domain of NLP. Finally, the syntactic PR algorithm used in this thesis will be detailed.

3.2 Methods of Pattern Recognition

This section will briefly cover the concept of Pattern Recognition (PR), and specifically the subdomains of Statistical and Syntactic PR. As statistical methods are the most common approaches to PR, it is important to explain how a statistical approach operates, and how it differs from syntactic methods. Afterwards, we will describe the principles of Syntactic PR so as to provide a context for the rest of the chapter.

3.2.1 Pattern Recognition

Pattern Recognition, in general terms, is the process of taking input data and labelling it in some way based on its presentation. The simplest approach is to manually define

a set of rules using which one can compare the input data against the training data and decide its label. However, attempting to do so against real data can become a large undertaking, both in the knowledge required within the domain to define well-formed rules, as well as the number of rules needed to sufficiently cover the problem. Most systems, thus, use techniques that are able to automatically build a model with which pattern recognition can be performed. In that sense, the type of model built depends greatly on the PR algorithm used. Particularly, the choice between two major classes of PR algorithms (Statistical and Syntactic PR) will greatly affect how the system operates. The decision between these two should be guided by the type of data being analyzed, as each excels at certain types of problems and performs poorly for others.

3.2.2 Statistical PR

The most commonly used form of PR is the Statistical variant. In this form, the “pattern” being labelled is a set of observations taken from the data called the “feature vector”. The analysts designing the PR system choose which features are to be considered, and depending on the features selected, it can greatly affect the accuracy of the PR algorithm. While the observations included in the feature vector can be numerical or nominal, the number of observations is, typically, predetermined by the expert. A major application of statistical PR is the classification of data. A classifier assumes that the data being processed belongs to one of a number of groups, or classes. The objective is to accurately classify each new data point into the proper group. The issue then, is to devise how the classifier is able to correctly determine this. Typically, data from the known classes already exists that somehow have their classes assigned. With this set, known as the “Training Data” set, a classifier can be trained with the existing data, so that it can correctly label new data. If there is no existing labelled data, or the amount available is insufficient, “unsupervised” methods can be used, which attempt to cluster similar data points to form unnamed classes, which can then be labelled after the data has been processed.

3.2.3 Syntactic Pattern Recognition

Statistical PR has the benefit of being fairly simple to use and having the ability to systematically create an optimal classifier. However, some problems cannot be properly depicted with a feature vector alone, and need a more complex representation for accurate PR. Examples of this include recognizing objects within an image and PoS tagging. These areas deal with structured data, where the entire “object” must be analyzed to properly make a decision. When using feature vectors to represent patterns, the vector will be of the same size and capture the same type of data, regardless of the structure or size of the pattern. While this is sufficient for many applications, one can easily comprehend why having a different approach for some cases is necessary, as the former results in a loss of information. Syntactic PR is preferred in these situations, as the representation used can scale to the size of the pattern, and properly represent the structural information adequately.

The basic idea behind Syntactic PR is that the pattern being processed is represented by a variable-sized set of *nominal* features, to avoid losing any structural information from the source. This allows for multiple types of representations, including hierarchical, relational graphs, and string-based representations. The rest of this chapter will cover each of these representations individually, starting with the hierarchical approach.

3.3 Syntactic PR: A Hierarchical Approach

This section will cover a common approach to Syntactic PR, where a hierarchical representation of a pattern is used for the purpose of classification. The hierarchical approach considers the problem being analyzed analogous to that of the syntactic parsing of a formal language. Because of this, it performs best when applied to patterns that have repeating sub-structures that compose larger patterns. For instance, the analysis of DNA could potentially be a PR application well suited for hierarchical analysis [33].

3.3.1 Constructing Representations using Formal Language Theory

A complex pattern can be simplified by breaking it down into smaller sub-patterns. By continuously breaking down a pattern, a hierarchical structure can be attained and analyzed based on its smallest constituent units. Instead of having to process the entire pattern at once, the structure can be analyzed using syntactic methods, utilizing formal language techniques.

As the pattern is to be treated as a string belonging to a formal language, some terminology must be defined. A grammar G is a set of rules used to produce a string that belongs to a language L . For the purposes of hierarchical syntactic PR, context-free grammars (CFG) can be used as a way to define the form that a pattern is expected to take. In a CFG, all rules take the form:

$$A \rightarrow b, \tag{3.1}$$

where A is a non-terminal node, and b is a string of terminal and/or non-terminal nodes. As the purpose of the CFG is to produce a string, the terminals are symbols from L 's alphabet, O . Using these rules, a complex pattern can be constructed from these primitive elements. The reverse is also true, and a complex pattern can be parsed using these rules, provided that the pattern belongs to the language defined. The image below shows how an input pattern string can be broken down using the rules of a CFG to constitute a hierarchical representation.

Context Free Grammar

$S \rightarrow NP VP$
 $NP \rightarrow Det Noun$
 $VP \rightarrow Verb NP$
 $Verb \rightarrow eats$
 $Det \rightarrow a \mid the$
 $Noun \rightarrow monkey \mid banana$

Example Parse

Figure 3.1: An example parse using a CFG, using a subset of English grammar. Lower case words indicate terminal symbols, while words beginning with Upper case indicate a non-terminal node.

3.3.2 Classification using Hierarchical Syntactic PR

In terms of Syntactic PR, a model can be defined to recognize patterns using a CFG. The smallest units that form the “alphabet” are referred to as the pattern primitives. A pattern primitive is an element that can be easily identified within the entire pattern, so that the more complex structures can be identified by building up from the primitives. The language of the pattern is called the “description language”, which consists of the set of primitives and a grammar.

Using a description language, a pattern can then be parsed using a variety of techniques. One of the most basic forms of parsing is the bottom-up approach, where the entire pattern is parsed starting with the initial primitive found, and then constructing the parse tree upwards until the starting symbol is found.

A major application for parsing a pattern based on a description language is classification. Assume that there is a description language for each potential class that a pattern could belong to. These description languages can then be used to parse an input pattern, and if a parse tree is successfully attained, it can be said that the input pattern belongs to the class associated with that description language.

Classifying in this manner requires that grammars be defined, but it is usually not the case that grammars for a specific classification problem have previously been constructed. Instead, the grammars must be built using patterns known to belong to certain classes. This process is called “Grammatical Inference”. With only examples of patterns that belong to a class, referred to as “positive examples”, the grammar inferred has a real potential of over-generating. Over-generation occurs when a sub-pattern in a positive example is discovered that causes the grammar to generate patterns that are not actually a part of the language. This would cause patterns to be potentially misclassified using the description language discovered. To combat this, grammar inference methods will also use “negative examples”, i.e., patterns that do not belong to the language, and that can be used to “tighten” the grammar rules discovered.

3.4 Syntactic PR Applications in NLP

This section will cover an instance of syntactic PR where a relational graph representation has been used in the field of NLP, in order to perform Word-Sense Disambiguation (WSD). As mentioned previously, WSD is the process of assigning the correct sense to a word, given the context that the word appears in. This is a very difficult task, as most words have several associated senses, sometimes having over ten. The obvious problem with WSD is that while context is needed to determine the correct sense of a word, the context itself also consists of words that must be disambiguated. As the correct senses of the input are interconnected, a syntactic PR approach would be well-suited for this problem, using relational graphs.

3.4.1 Overview of the Structural Semantic Interconnection Algorithm

This approach, used in the research of Navigli and Velardi [26], creates the Structural Semantic Interconnection (SSI) algorithm, which is an approach that uses Syntactic PR to disambiguate word senses. As mentioned previously, a sense is the potential meaning of a word. Typically, a word has multiple potential senses associated with it, but the correct sense can usually be understood by a human based on common usage and the present context. In order for a computer to also interpret the correct sense, it must be capable of understanding the context, i.e., specifically of how nearby words relate to the current word being considered. The relational graphs used are meant to connect related senses, so that a large net of senses can be constructed from the context. As the known context grows, it should become easier to find the appropriate sense of a word that is related to the senses already known.

To begin with, a knowledge base of word-senses needs to be acquired in order to construct the relational graphs. Existing sources, namely WordNet, are used for this purpose, as it is, in itself, an extensive database of word-senses and relations (especially for nouns). As WordNet is the primary source of the SSI, nouns also constitute the main focus of the experimentation. These word-senses and relations are extracted to form “semantic graphs”, which are labeled directed graphs like the one shown below.

Figure 3.2: A simplified semantic graph displaying the basic concept of the relational graphs constructed. In reality, each node would branch off with numerous relations, with a maximum distance of three.

The central node is the word-sense under consideration, and the graph expands to three nodes from the center. Various sizes were experimented with, and the number three was determined to yield the best results. By capturing the surrounding related nodes, two senses can be compared to determine if they are closely related, as they may have coinciding nodes in their semantic graph.

3.4.2 The SSI Algorithm

The SSI algorithm takes as its input a list of terms, T , that are considered to be in the same context. The goal of the algorithm is to assign the correct sense to as many of the terms as possible. The WSD operates by primarily analyzing the surrounding context of a word in order to determine the most likely sense. However, when given a list of terms as its input, at least one word must be initialized with a sense in order for the other words to have a context. If there exists a term t that has only a single sense

associated with it, this process is simple. Otherwise, a more complex initialization stage is required. Once the first sense is determined, it is possible to disambiguate the other terms in T with the context that has been provided. However, as more words become disambiguated, the collective semantic graph will cover a larger area of meaning. As it expands, the algorithm will become more capable of determining the senses of the remaining words, causing it to iteratively disambiguate word-senses as more contexts become available. Due to this, the SSI algorithm has two main steps: The initialization step and the iterative step.

The initialization step searches T for any words that have only a single sense. If none are found, the word with the least number of senses is chosen and the algorithm branches off, with each potential sense acting as the initial sense used. Note that this is only what the algorithm does in its general usage. During experimentation, however, where the actual results were calculated, gold standard senses were used and the initialization step unambiguously chose the correct sense for the initial term. This is because the challenge essentially consisted of disambiguating the words in the definition of a word-sense, and so, the word-sense in question was already disambiguated.

For the iterative step in the algorithm, a set P is maintained that contains the terms that have not yet been disambiguated. For each iteration, if no terms have been removed from P , the algorithm terminates. In each iteration, any term t in P that has a correlated sense with the semantic graph of I is selected. The sense of t is represented as S , and the correlated sense found in I is represented as S' . The likelihood that this sense is the correct interpretation of t is calculated by the following formulae.

$$f_I(S, t) = p(\phi(S, S') | S' \in I) \quad (3.2)$$

$$\phi(S, S') = p'(w(e_1 \cdot e \dots \cdot e_n) | S \xrightarrow{e_1} S_1 \xrightarrow{e_2} \dots \xrightarrow{e_n} S_n), \quad (3.3)$$

where e is an edge in the semantic graph, and $w(\dots)$ is the sum of the weights of each connection in the semantic graph. f_I , p' is the average sum of weights for each path

between S and S' , and $p(\dots)$ is the average sum over every S' in I that S connects to. If the likelihood measure f_I is greater than a set threshold, then the sense is accepted and entered into I .

3.4.3 Experimental Results

As mentioned previously, this algorithm has been evaluated using a specific challenge, i.e., the WordNet gloss disambiguation task. In WordNet, a word-sense has a brief definition called the “gloss”. To evaluate the WSD algorithm, a selection of these glosses were tagged with their correct senses. For the SSI, the initialization consisted of populating I with the initial sense of the gloss. The list of pending terms P was then populated with a selection of words from the gloss. As the SSI was tailored towards nouns, the SSI was evaluated by both including non-nouns in the gloss and by excluding them.

It was found that when the size of T increased, the algorithm became more accurate, due to the fact that there was a larger context available for the disambiguation of subsequent words in the iterative cycle. Because of this occurrence, the words from the gloss of the hypernym of the word-sense being considered were also added to T and P . The image below shows how the algorithm was initialized for a given word-sense.

Figure 3.3: The words selected for the context when working with the nouns of the word-sense “glass”. The nouns in blue are from the actual word-sense, and are the words that are used to measure the success of the algorithm. The words in red are from the hypernym “container”, and are not used in measuring the success of the algorithm, but in providing additional context.

The chart below shows the results of the WordNet gloss disambiguation task.

Table 3.1: The results show a high level of precision in disambiguating nouns, at the expense of recall. The attempted percentage refers to how many senses were able to be assigned using the SSI algorithm.

	Nouns	Verbs	Adj.
Precision	86.0%	69.4%	78.6%
Recall	44.7%	13.5%	26.2%
Attempted	52.0%	19.5%	33.3%

The findings demonstrated that the system was able to achieve high levels of

precision when dealing with nouns, but with a low recall. The low recall is partially due to the threshold mechanism that prevents sense relations that were discovered but that had low confidence from being added to the set I . Removing the threshold had the potential to raise recall at the sacrifice of precision. A notable effect that this algorithm had, was the ability to produce an exact path between the senses chosen, allowing for analysts to see why exactly certain senses were selected, and how all of the senses in the context were related.

3.5 Information Theoretic Syntactic PR

As mentioned previously, this thesis uses Syntactic PR methods to assist in MT. The approach that we invoke is different from the ones covered earlier in this chapter, because a string-based representation is used rather than a heavily-structured one. We utilize research performed by Oommen and Kahsyap [28] to develop the Information Theoretic Optimal Pattern Recognition (OptPR) algorithm to achieve this. The OptPR algorithm has two major modules: a dictionary of strings, and a model of a channel that arbitrarily adds noise to a string. It operates by assuming that the input pattern to be recognized is one of the entries in the dictionary, garbled after passing through the noisy channel. Using the channel's model, the likelihood of the input pattern being a certain entry in the dictionary can be calculated, so that the dictionary entry which is most likely to be the original string, can be determined.

3.5.1 The OptPR Model

The first component, the dictionary, is a set of example patterns against which the input has to be matched. For example, if the PR problem involves matching noisy English words with ungarbled, actual English words, the dictionary would consist of the actual words. Both the original and noisy version of the strings must consist of elements contained within the alphabet A , in this case the English alphabet.

To model the noisy channel, three distributions are used: the quantified insertion distribution G , the qualified insertion distribution Q , and the substitution/deletion

distribution S . Before describing how these three are used in the model of the noisy channel, each will be individually described. The original string from the dictionary is referred to as U and the noisy version of the string as Y .

The quantified insertion distribution G covers the set of positive integers. For a positive integer z , $G(z)$ is the probability of z insertions occurring when U passes through the noisy channel. $G(z|U)$ is then the probability of z insertions given that the string U is passing through. To simplify things, z is assumed to be independent of U .

The qualified insertion distribution Q covers each element of the alphabet being used. For a given entry in the alphabet a , $Q(a)$ is the probability of a being inserted during the garbling process.

The substitution/deletion distribution S gives the conditional probability distributions over $Ax(A \cup \lambda)$ where λ represents a null character. $S(b|a)$ is the conditional probability that the element a in U is mutated into the element b . If the element b is an element of the alphabet, it represents a substitution. If b is the null symbol λ , it represents a deletion.

Using these three distributions, the noisy channel can now be described. First, a random number of insertions is chosen using the distribution G . Once a number is attained, the positions in which the insertions will occur are chosen, with each potential position in U being equally likely. When the positions are selected, they are filled by randomly selecting an element in A using the distribution Q . Afterwards, the original elements of U are randomly substituted/deleted using the distribution S . Once this is complete the λ symbols are dropped and the string Y is attained. The image below shows the basic process of adding noise to a string U .

Figure 3.4: The three distributions of the model are used to generate a noisy string Y from U .

The phenomenon of modeling using a “noisy channel” has, of course, been used in the prior SMT art. However, the channel used in previous SMT systems operates somewhat differently from the Syntactic PR channel model depicted above. We shall now clarify how a noisy channel is invoked in SMT systems.

Consider Figure 3.5 which depicts a generalized noisy channel model used for SMT [21].

Figure 3.5: The language model generates a TL sentence, e , that is transmitted through the channel’s translation model into the SL sentence, f . The original sentence e is then estimated with a decoder function obtaining $\hat{e}$.

The basic idea behind the SMT noisy channel approach is that it assumes that the SL sentence being translated, f , originally belonged to the TL. In order to obtain this TL sentence e , it invokes a language model to generate sentences. This is done

because it is not feasible to collect every potential sentence belonging to the TL. The original TL sentence is assumed to have been transformed into the input by it having passed through the translation model. Finally, the original sentence in the TL is estimated using a decoder, yielding a translation $\hat{e}$.

With regard to functionality, the translation model of the SMT noisy channel can be seen to be similar to that of the OptPR noisy channel. It is, however, difficult to compare the two as there are a large number of different implementations for the SMT translation model component. In any case, the essential purpose of the translation model is to estimate the combined probability of a sentence e becoming a sentence f given every potential alignment between the two sentences. On the other hand, the OptPR noisy channel estimates the total probability of a string U becoming a string Y by means of all potential sequences of elementary edit operations. These edit operations are performed by invoking the three distinct distributions depicted in the figure describing the OptPR noisy channel. This, combined with the fact that a dictionary is used in place of a language model, allows us to optimally and “totally” search the problem space during the decoding phase.

3.5.2 The OptPR Algorithm

As stated, in terms of PR, the input pattern is considered to be a noisy version of a dictionary entry. The input pattern is thus considered Y , and each entry in the dictionary is potentially U . The OptPR algorithm uses the three distributions of the noisy channel model to calculate $Pr[Y|U]$, the probability of attaining Y given the original string U . By calculating this probability, the input pattern can be matched with the dictionary entry with the highest probability. The algorithm to calculate this is shown below.

Algorithm EvaluateProbabilities

Input: The strings $U = u_1u_2 \dots u_N$, $Y = y_1y_2 \dots y_M$, and the distributions G , Q and S .
Let $R = \text{Min}[M, N]$.

Output: The array $W(i, e, s)$ for all permissible values of i , e and s and the probability $\text{Pr}[Y|U]$ as defined by equation (7).

Method:

```

 $W(0, 0, 0) = 1$ 
 $\text{Pr}[Y|U] = 0$ 
For  $i = 1$  to  $M$  Do
     $W(i, 0, 0) = W(i - 1, 0, 0) \cdot Q(y_i)$ 
For  $e = 1$  to  $N$  Do
     $W(0, e, 0) = W(0, e - 1, 0) \cdot S(\lambda|u_e)$ 
For  $s = 1$  to  $R$  Do
     $W(0, 0, s) = W(0, 0, s - 1) \cdot S(y_s|u_s)$ 
For  $i = 1$  to  $M$  Do
    For  $e = 1$  to  $N$  Do
         $W(i, e, 0) = W(i - 1, e, 0) \cdot Q(y_i) + W(i, e - 1, 0) \cdot S(\lambda|u_e)$ 
For  $i = 1$  to  $M$  Do
    For  $s = 1$  to  $M - i$  Do
         $W(i, 0, s) = W(i - 1, 0, s) \cdot Q(y_{i+s}) + W(i, 0, s - 1) \cdot S(y_{i+s}|u_s)$ 
For  $e = 1$  to  $N$  Do
    For  $s = 1$  to  $N - e$  Do
         $W(0, e, s) = W(0, e - 1, s) \cdot S(\lambda|u_{s+e}) + W(0, e, s - 1) \cdot S(y_s|u_{s+e})$ 
For  $i = 1$  to  $M$  Do
    For  $e = 1$  to  $N$  Do
        For  $s = 1$  to  $\text{Min}[(M - i), (N - e)]$  Do
             $W(i, e, s) = W(i - 1, e, s) \cdot Q(y_{i+s}) + W(i, e - 1, s) \cdot S(\lambda|u_{e+s}) + W(i, e, s - 1) \cdot S(y_{i+s}|u_{e+s})$ 
For  $i = \text{Max}[0, M - N]$  to  $M$  Do
     $\text{Pr}[Y|U] = \text{Pr}[Y|U] + G(i) \frac{N!i!}{(N + i)!} W(i, N - M + i, M - i)$ 

```

END Algorithm EvaluateProbabilities

Figure 3.6: The algorithm calculates the probability of string Y being derived from string U , based on the distributions Q , G , and S . The algorithm operates in cubic space and time.

The algorithm essentially calculates the combined probability of every potential combination of operations that would cause U to transform into Y . For the purposes of string matching, the input string Y is compared to every string in the dictionary, and the one that returns the highest probability of $P[U|Y]$ is selected as the correct match.

3.5.3 Experimental Results

Previously, an example was given in [28] which tested the accuracy of OptPR where noisy English was used as an input pattern to compare against ungarbled English dictionaries. This scenario was utilized to yield the experimental results that showed the power of the algorithm. 342 words were used to comprise the dictionary, and two sets of 1026 noisy strings were generated using the noisy channel model.

The model itself was defined as follows: Q was described as a geometric distribution, G was a uniform distribution, and S was defined using the likelihood of a key being mispressed on a standard QWERTY keyboard. Examples of noisy strings are shown below:

Table 3.2: The table shows how noise caused by the model can greatly affect the string U , and how powerful the algorithm is in being able to recognize the original string in spite of this noise.

Original Word	Noisy Word	Num. of Errors
accomplishments	accoplsuments	3
beginning	ssbehsimgjnmmg	10
control	fodvntopl	5
development	dbvelrlpmenr	4
strength	mzeckieeotrenxsbt	13

The experimental results, assuming that the three distributions for the model are known, are shown in Table 3.2. With this experiment, the model is already known as it was used to create the test data. However, in a real-world problem, when dealing with the PR of unknown distributions, this model must be estimated instead. To account for this, percentages of error were introduced into the distributions in order to simulate an estimated model, and how well the algorithm performs with respect to it.

Table 3.3: The following table shows how accurate the OptPR algorithm is in string matching with both the model used to create the strings, as well as with variations of the model with percentages of error added to it.

0% Error	5% Error	10% Error	15% Error
97.66%	97.66%	97.47%	97.37%

As can be seen, the algorithm is highly accurate if the distributions describing the model are known. And while this accuracy decreases marginally at the 10 and 15 % levels of error, the metric itself is still high when compared to other string matching methods.

3.6 Conclusion

Beyond the fact that this project focuses on the concept of using syntactic PR methods in MT, the decision to use syntactic PR methods over statistical is essentially based on the fact that sentences are not of fixed length. It is commonplace for MT systems to work with a block of text by processing each sentence of the text individually. For some aspects of MT, it is sufficient to use statistical PR methods. But when the whole structure of the sentences needs to be represented, syntactic PR is better suited for the problem.

The specific representation used will be a flat string. This is because we will be working with the English and Japanese languages to perform string matching between the two. While English can be parsed to form a hierarchical structure, Japanese syntax has a flat representation. Due to this, it makes sense to use string-based representation to allow for easy comparison. It then follows that the OptPR algorithm be used in our approach, as it has been shown to excel in string-based pattern matching.

Chapter 4

Contributions: Tagging Aspects

4.1 Introduction and Motivation

4.1.1 Chapter Overview

This chapter will discuss the contributions that this thesis makes to the tagging aspects of NLP, specifically in the field of MT. As discussed previously, the majority of translation systems have an analysis phase that occurs *prior* to the actual act of translation. The purpose of the analysis phase is to extract information from the input text that is not already explicitly available, and to present the input in a way that is useful for translation. Tagging is a process that occurs during this phase that assigns tags to segments of the input text in order to communicate additional linguistic data. PoS tagging is the most commonly used tagging process, where individual tokens are assigned tags based on their parts-of-speech. PoS tags are vital in performing MT, as many later processes in analysis, translation, and synthesis require this information. One such use of this information is the ability to represent the syntactic structure of an input sentence using these tags. An example of a representation is a constituent parse, which represents English sentences as trees, with the PoS tags acting as leaves from which the tree can be built up. This thesis uses a flat representation for a sentence's syntax rather than a tree, so that string-based pattern recognition algorithms can be used to facilitate translation.

4.1.2 Context

As we already have mentioned, the language pair being used in this thesis is English-to-Japanese. As such, a parallel corpus of these two languages is being used as the primary data source. The data gathered here will be used to build a translation model and a dictionary to be used in the pattern recognition algorithm (discussed in Chapter 5). As the model of translation is purely syntax-based, sentence pairs in the corpus that are only *semantically* equivalent are disregarded. The raw sentences contained in the corpus are converted into flat syntactic representations to be used both in building the model and at runtime, combining SMT and EBMT techniques.

4.1.3 Purpose

The purpose of this approach is to evaluate the possibility of using syntactic representations as a viable method for translation between two structurally different languages. Naturally, a language will have less unique sentence structures than instances of sentences in a sufficiently large corpus. Additionally, there should at least be a subset of language-to-language sentence pairings in a parallel corpus in which similarly-structured sentences in language S correspond to sentences in language T . By pairing phrases between such sentences in the corpus, repeating structures can be discovered in the existing data to aid in the translation of new data. This allows for a syntactic-transfer model to be built using less data than a typical SMT.

4.1.4 Rationale

A benefit of using this approach is the mitigation of the need for large corpora to build translation models. A translation model outside of interlingual methods is typically specific to a language pair. For an SMT, this would mean acquiring a parallel corpus between each language in the translation system that is sufficiently large to produce a passable model. By building models using sentence structures rather than raw sentences, the size of the parallel corpus can be reduced.

4.1.5 Roadmap

The following section lists the various contributions that this thesis makes in terms of the tagging aspects of MT, as well as covering the previous research that it builds upon. Thereafter, the purpose of each individual contribution is given, as well as the details of the approach used.

4.2 Contributions of this Chapter

The major contributions of this thesis, in regards to the tagging aspects of MT, are listed in this chapter. First, previous research related to the contribution will be summarized, followed by the contribution itself and how it builds on the previous research. The contributions in this section are: The usage of template sentences in MT, the tagging methods employed to acquire the TSs, the ways in which the tags were simplified, and the phrase-level parsing of the TSs.

4.2.1 Contribution: Template Sentences

The research performed by Yamada and Knight [41] in their SMT for English-Japanese translation used constituent parses as the representation of the English input sentences. The constituent parse nodes used PoS tags as labels, and the translation model was trained to perform reorder and insertion operations on parent-child nodes to mold the English syntactic structure into a Japanese structure. The usage of TSs builds on this research by also representing the English input text with the syntactic categories of the words used, rather than the words themselves. The major difference in these two methods is that TSs are used in a string-based pattern recognition algorithm, and so are adapted to have a flat representation. The downside of using a string-based representation in this case is that the translation stage loses the ability to perform major word migrations that comes naturally to tree-based reordering operations. The benefit to using a string-based representation, however, is that the issue of building a model for the pattern recognition algorithm becomes an issue that is very similar to that of building the translation models of SMTs.

An SMT using purely statistical methods of translation requires that two models be built before the translation can be performed. One of these models is the translation model, which gives the probability of a word in the TL becoming a word in the SL. A parallel corpus is used to build this model, but the corpus used usually does not contain word-to-word mappings that are required to accurately calculate the probability of word translations. The IBM Models 1-5 handle this issue by iteratively building a translation model, performing mappings on the parallel corpus, and then rebuilding the translation model. The parallel corpus used here has its raw sentence converted into the TS representation, but a translation model of some sort is required to approximate the likelihood of a TL tag being translated into an SL tag. The IBM Models can be used to acquire this model, as it is essentially the same problem faced when building a standard SMT.

4.2.2 Contribution: Tagging Methods

PoS tagging is a frequently-used method of acquiring linguistic data used in most NLP projects. The usage of PoS tags varies greatly depending on the objective of the project in question, and often the tagset used must be modified to better suit their needs. One method, called “PoS Tag Adaptation” uses a generic tagger to generate training data on a sub-domain, and combines it with grouping morphologically similar words, and thus attempts to cluster word groupings to discover word-classes that are suitable for the sub-domain being worked with. This approach is well-suited to working with subdomains of a single language, but this thesis requires the adaptation to a tagset covering the *general* usage of the two languages. As such, two language dependent taggers are used as input, and similar tags for content word classes are combined, to form a shared tagset.

4.2.3 Contribution: Tag Simplification

In order to reduce the complexity of their translation model, Yamada and Knight simplified the constituent parses used for English input prior to performing syntactic translation. Constituent parses left as-is only use PoS tags to label the leaf nodes of

the trees, and for higher level nodes use structural labels such as “VP” (verb phrase). All constituents in a parse have a so-called “head word”, which is the word that all other elements of the constituent modify. To simplify the labelling of the parses, they replace all labels in the tree with the PoS tag of the head word for each constituent. The tags used for this thesis already consist of PoS tags (excluding the Japanese particles), but certain simplifications can still be performed. There are occurrences of several tags being dependent on one, such as the particles that follow a Japanese verb to indicate tense. In some cases, these tags can be combined with the “head tag”, and then re-integrated after the translation phase is done.

4.2.4 Contribution: Phrase-Level Template Parsing

The usage of a parallel corpus with pre-processed sentences builds on the research concerning EBM systems performed by Nagao [25]. Nagao’s initial research suggested the use of phrase-level units of sentences to map equivalent structures between languages, for use in translation. This concept was then implemented by Nagao and Sato, using dependency parse sub-trees rather than phrases. This decision allowed for the entries of the parallel corpus to be mapped at different levels of granularity, for the purpose of reusability. Our research returns to the usage of phrase-level translation units, instead segmenting the parallel corpus sentences into the verb, subject and object components of the raw sentence. This allows for the major word ordering distance between English and Japanese to be handled in a post-processing phase, while still maintaining a string-based representation.

4.3 Template Sentences

A Template Sentence (TS) is a representation of a sentence that replaces some or all of the tokens with placeholders. The placeholders represent, in some way, a generic form of the word or phrase that it replaces. In this case, the placeholders that are used are the syntactic word classes of the original tokens. TSs are used to represent the parallel corpus in both building the translation model as well as in creating the

dictionary to be used in the translation algorithm.

4.3.1 Purpose of Contribution

The main purposes of using TSs in translation are to reduce the data needed to build a translation model as well as in creating a translation system that focuses on forming syntactically correct sentences in the TL. By requiring that the source text be matched with a TS that occurred in the parallel corpus, the system guarantees that the output of translation will correspond to a sentence structure that has occurred in the TL.

4.3.2 Description of Contribution

A sentence can be seen as a sequence of words and symbols arranged in a particular manner. In that sense, a TS is the same as a regular sentence, except that some or all of the words are replaced by placeholders. These placeholders can be replaced by various instances of words while still forming a grammatically correct sentence.

Words can be classified in various ways according to their meaning. For instance the words “apple” and “grape” can be classified as words referring to an instance of “fruit”. In the sentence “He ate the apple.”, the word apple can be replaced by most instances of fruit. The TS “He ate the [fruit].” allows for the structure to be reusable for various sentences.

Words can also be classified according to their syntactic function, as seen with PoS tagging. Rather than classifying “apple” as a fruit, PoS tagging would assign the tag corresponding with the word class “noun”. The TS example used previously would then be “he ate the [noun].”. As the classification is purely syntactic with no regard for meaning, this template can be used to generate a large number of “nonsensical” (or meaningless) sentences, most of which will, at least, be syntactically correct. Rather than replacing a single word with a placeholder, the TSs used in this thesis replaces every word with a tag, so that sentences are represented purely by their syntactic structure. The figure below shows how a TS that is fully comprised of tags, can be used to generate different sentences.

Figure 4.1: Replacing the words of a sentence with their syntactic classes results in a “Template Sentence”, which can be used to form different sentences with the same structure.

In addition to generating similar sentences in the same language, TSs can also be used to perform various levels of translation between two languages. The simplest example of this would be that of having a single word in a sentence being replaced by a placeholder in the two respective languages. The word being added to the sentence in the SL then only needs to be translated and placed into the matching sentence in the TL. Multi-word translations can also be performed using this same process, but some sort of mapping operation is needed so that the input words in the source text can be correctly placed in the TL sentence. The image below shows the concept involved in such a translation using templates, demonstrating the additional complexity of performing multi-word translation.

Figure 4.2: Single and Multi-Word translation can be performed using templates, as long as a mapping function exists.

A major issue in using TSs in translation is thus the need to “invent” such a mapping function between the two languages. This is a problem similar to that of SMT, where the translation model is built using a parallel corpus with no word-to-word mappings. The same algorithms used in SMT can then be used with TS translation, except that instead of using the raw parallel corpus to build the translation model, every entry in the corpus is replaced by its TS equivalent. Assuming that this model can be used to discover the most probable TL template in the parallel corpus given the input text, the TS can be used to facilitate translation, as seen above. As it is a complete sentence being translated rather than simply just content words, the mapping becomes more complex than just replacing words of the same class, as demonstrated in the image below.

Figure 4.3: When mapping outside of the content classes, words are often mapped to different classes or simply dropped/added.

As can be seen, the more complex a template becomes, the potential for errors also increases. By reducing the complexity of the TSs used, the accuracy of the mapping algorithm should, ideally, increase. The risk of reducing the associated complexity is the potential loss of information. As an extreme example, if all TSs were simplified to the singular placeholder S , representing a sentence, TSs in the SL and TL would be correctly matched every time. Obviously, such a mapping is useless as it provides no additional information. On the other side of the spectrum, if every word in both the TL and SL had its own specific placeholder, all information from the original sentence would be preserved, as it and the TS would be identical. Using such a mapping, however, would lose any potential benefit that TSs could offer. The goal is then to achieve a middle ground between these two extremes so that the accuracy of the mapping algorithm can be maximized while preserving as much information as possible.

Since the word-to-word-class is a one-to-many relationship, the number of unique TSs should be smaller than the number of unique sentences in a sufficiently large

corpus. The reason why they are fewer in number is that a single TS can potentially represent several sentences within the corpus. Additionally, a TS is capable of representing sentences outside of the corpus used. As such, TSs should represent a larger scope of a language than raw sentences in two corpora of the same size. A very large parallel corpus is needed to generate competent translations using SMT methods. Since translation models are 1:1 in terms of languages, a large parallel corpus is then needed for every possible language pairing in a translation system, unless pivot techniques are used (which reduces the quality of the translation). Using TSs mitigates the need for large corpora, allowing for translation models to be generated between languages where linguistic data is more sparse.

Beyond sentence-to-sentence translation, TSs can also be used for phrase-level translation. If sentences in a parallel corpus can be broken down into corresponding sub-structures, translation can be performed by breaking down the input sentence into sub-structures, translated, and then rebuilt into a complete sentence in the TL. The benefit to doing this would be that the TL sentence structures that don't appear in the parallel corpus can be generated. The downside is that, since the sentence structure obtained is not guaranteed to have occurred in the original corpus, the sentence structure is potentially invalid.

4.4 Tagging Methods

While the previous section covered the concept of TSs, in this section we detail the actual acquisition of the templates. By using two language-dependent taggers and then unifying their results, we acquire a shared tagset that is used to label the raw sentences of the parallel corpus. These labels are then used to build the base TSs to be used for translation.

4.4.1 Purpose of Contribution

The tagsets used by the language-dependent taggers were designed specifically to represent their respective languages, and not to facilitate the task of translation into

other languages. By creating a new tagset based on the results of these taggers, it can be specialized specifically for the domain of MT. By unifying the tagsets used in building TSs, we can emphasize the differences between the languages when the translation model is built. By combining the content word classes between the two languages, it leaves the syntactic word classes and particles to be mapped between the two languages.

4.4.2 Description of Contribution

As mentioned in Chapter 2, PoS tagging for the English language is a very reliable process, yielding more than 98% accuracy. The most common set of tags used for the English Language is the Penn Treebank tagset, and so, this set has been used for generating the initial templates on the English side.

To tag sentences in the Japanese language, we have utilized the ChaSen tagger. The tags used by this system are very different from the Penn Treebank tagset, as each set contains tags designed specifically for their respective language. One important tag used by the ChaSen tagger is the “particle” tag. As Japanese sentences have a free word order, particles are needed to mark the various subjects and objects of a sentence. As this information is important in forming a syntactically correct sentence, these particles are preserved in the TSs extracted, rather than replacing them with the tag “particle”. The figure below shows the process of converting a Japanese sentence into a TS, while preserving the particles. It also displays how two templates of the different languages become comparable.

Watashitachi wa byoin ni ikimashita

Pronoun wa Noun ni Verb kimashita

Figure 4.4: The figure shows how a Japanese TS can be acquired using the ChaSen tagger. Due to the syntactic importance of Japanese particles, they are preserved in place of their tags.

The templates extracted, left as-is, are not easily comparable. Even for word classes that are common to both tagsets, the two taggers give sub-classifications that add a layer of complexity that is unnecessary for the purpose of a TS. Table 4.1 displays a large number of tags associated with the base word-class of “Noun” assigned by ChaSen, the Japanese tagger, along with a brief description for each associated tag. The English translations for the tags were acquired from the jaSlo project page [8], and these have been derived from research conducted to generate a Japanese-Slovene dictionary [7].

Forcing the Japanese TSs to use the tags “NN” and “NNS” in place of its various subclasses would resolve this ambiguity. The solution of converting the Japanese tagset to the Penn Treebank tagset works in this case, but there are several tags in each language’s tagset that refer to classes that have no equivalence in the “alternate” language. Any conversion of these word classes would result in a loss of syntactic information needed for the template sentences to be valid.

Rather than forcing all ChaSen tags to be converted, we opt to perform a selective conversion. More specifically, content word classes (noun, verb) are converted into Penn Treebank tags, while the particles preserved previously remain in the final TS. This allows for the shared aspects of both languages to be more easily comparable. The figure below shows a Japanese template partially converted, resulting in a more comparable form to its respective English counterpart template.

Table 4.1: Due to the various sub-classifications of the word-class “noun”, it becomes difficult to compare sentence templates. More specifically, the Japanese tagset has a large number of subclasses that are unnecessary for the purposes of this study. In contrast, the Penn Treebank tagset only contains two noun tags, namely “NN” (singular noun) and “NNS” (plural noun). This table mentions the so-called “bound nouns”, referring to words that rely on another word in the sentence, as well as “-na” adjectives, a special form of adjectives found in the Japanese language.

Japanese Noun Tags	Description
Noun(Verbal)	Noun used with an irregular verb.
Noun(General)	Standard usage of noun.
Noun(Numeral)	Noun used as a numeral.
Noun(Bound, General)	Bound noun used with a general noun.
Noun(adjective -na)	Noun modified by a “-na” adjective.
Noun(adverbial)	Noun modified by an adverb.
Noun(Bound, Auxiliary)	Bound noun used with an auxiliary verb.
Noun(+nai)	Noun modified to be negative.
Noun(Bound, Adverb)	Bound noun used with an adverb.
Noun(Bound, adjective-na)	Bound noun used with a “-na” adjective.
Noun(Conjunction)	Noun used as a conjunction for other nouns.
Noun(Verbal, Bound)	Bound noun used with a verb.

Watashi wa byoin ni ittekimashita.

Figure 4.5: This figure shows that by selectively converting ChaSen tags into Penn Treebank tags, a Japanese TS can become comparable to an English TS.

4.5 Tag Simplification

Tag Simplification is the process of simplifying both singular tags and tag sequences that occur in TSs. In the context of translating from English to Japanese, there are cases where different tags in English are consistently used to represent the same tag in Japanese. These different tags are simplified into the same tag, leaving any information lost in the process to be re-added in a post-processing phase. As oversimplification of tags can lead to a great deal of information loss that cannot be reliably retrieved, the simplification of tags is left purely to instances where the information can be re-attained procedurally, without error. Certain tag sequences can also be treated in a similar manner, where a sequence of tags is simplified into a single tag and then expanded later.

4.5.1 Purpose of Contribution

The main purpose of tag simplification is to reduce the complexity of both the TSs used for training and translation, as well as in reducing the number of unique tags in the tagset. Simplifying the complexity of TSs and the tagset has the potential of increasing the accuracy of translation as seen previously, as long as no information becomes irretrievable in the process. Additionally, the overall performance can be improved, because a smaller alphabet and dictionary reduces the time it takes for the pattern recognition algorithm to find the most probably TL template.

4.5.2 Description of Contribution

The first simplification that can be performed is the grouping of certain tags that achieve a similar syntactic role. An example of this is the grouping of the English tags “NN” and “NNS”, simplified to become merely the “NN” tag. As the information lost by this simplification can be stored and applied in a separate post-processing stage, there is no risk in applying this simplification.

Another form of simplification is obtained by reducing multi-tag sequences into a single tag. Japanese verbs are typically followed by various particles to apply a tense

to the base form. These particles can be removed from the TS and have that portion of the verb phrase be represented simply by the “VB” tag. Since Japanese is the TL, the tense of the verb will be provided by source text, which can then be applied to the TL verb after the translation phase.

The image below shows both the instances of simplification described above being applied, along with the post-processing stage that can undo the simplification.

Figure 4.6: By simplifying the templates in ways that allow information to be reinserted, the performance of the mapping algorithm is potentially increased.

This simplification reduces complexity of the translation system in two ways. First, it reduces the size of the template sentences used to build the translation model, which we have seen previously has the potential of increasing its accuracy. An additional benefit of this simplification is that it reduces the number of unique TSs to be stored in the TL dictionary. As the pattern recognition algorithm compares against every entry in the TL dictionary, this simplification reduces the time it takes

for translation. Secondly, as certain tags are no longer used due to the simplification, the size of the tagset is also reduced. With less tags (provided they are not over-generalized) it will take less data from the corpus to build an accurate translation model.

4.6 Phrase-Level Template Parsing

Phrase-Level Template Parsing refers to the process of dividing the acquired template sentences into sub-sentence units. There are various ways in which a sentence can be divided into sub-structures, and for the purposes of this project the template sentences are divided into their Subject-Verb-Object components. Depending on the language of the sentence being analyzed, these components are extracted in different ways. For English, the structure of a well-formed sentence can typically be used for the parsing, while Japanese requires looking at the particles to determine the subject, verb and object of a sentence.

4.6.1 Purpose of Contribution

The purpose of extracting these phrases from the template sentences is to mitigate the word ordering difference between the English and Japanese languages. As mentioned previously, a major challenge in translating between this language pair is the distinct difference in sentence structure. Particularly, the general word order of English sentences being Subject-Verb-Object, versus the Japanese sentence structure of Subject-Object-Verb, requires that major migration of words be used when performing a syntactic translation. As the algorithm being used to translate uses string-based methods, this type of re-ordering is not natively supported, and so this must be handled in a separate stage. By segmenting these components and then handling them separately, the algorithm can translate the individual segments, and then the re-ordering operation can be performed afterwards. Additionally, the structure of these components differ greatly from each other, and so separating and then building a model specifically for the verb, object, and subject phrases, allows for a more

specialized algorithm for translation. The additional benefit to this, as mentioned previously, is the ability for sentence structures that are not directly represented in the corpus, to be built. The image below shows two TSs in the TL dictionary being used to construct a sentence with a unique structure.

Figure 4.7: Using phrase-level matching for TSs allows for sentence structures that are not covered by the corpus to be generated.

4.6.2 Description of Contribution

The structure of English and Japanese sentences is very different, and so the acquisition of the verb, subject and object must be performed in a manner unique to each language. For English, a constituent parse can be used to identify these three sub-structures. As for Japanese, the subject and object can be ambiguous in certain situations. We employ a safe approach here, i.e. one that accepts Japanese sentences that contain particles that clearly denote the subject and the object of the sentence. A side effect of this extraction process is that it filters out sentences within the corpus

that are not well formed. Doing so helps limit the translation model to only be built from well-formed sentences, which is important for syntactic translation.

The next step is to bind the extracted phrases to the phrases of its parallel sentence. Ideally it would be as simple as binding the verb, subject, and object of each sentence pair together, but complications can arise in doing so. The main issue is whether the sentence pairs being considered are semantic or syntactic translations. If it is a purely semantic translation, then the structure of the two sentences could be completely arbitrary. A second issue is the possibility of the extraction methods failing to correctly extract the three phrases. Sentence pairs affected by either of these issues must be filtered out in order for the training data that is used to be as correct as possible. This is accomplished by comparing the content words of the paired phrases to see if the number and senses of the content words match. To compare senses, we utilize the Princeton WordNet and the Japanese WordNet. By comparing the potential senses of each content word between the phrases for an approximate match, we can confirm whether both the extraction process succeeded and if the sentence pairs are syntactically equivalent. Once the filtering has been completed, we obtain a set of sentence pairs, segmented into phrases for both the training of the translation models, and for building the TL dictionary.

Chapter 5

Contributions: Enhancements Using Syntactic PR

5.1 Introduction and Motivation

5.1.1 Chapter Overview

This chapter will cover the aspects of this project that are related to Syntactic PR, and more specifically how it can be used to enhance MT systems. While the previous chapter covered how the parallel corpus has been processed and transformed into the TS representation, this chapter will show how these TSs will actually be used to build a model, which can then be utilized during the translation process. Additionally, the process of using the OptPR algorithm for string-based comparisons between the languages will be described.

The specific aspect of translation that we hope to improve, is the capability of a translation system in choosing a syntactically-correct sentence structure, when transferring into the TL. By running the input text against a dictionary of sentence structures extracted from the parallel corpus, we can, at least, guarantee that the tag sequence chosen has occurred in the TL. This tag sequence can then be used as a template for constructing the final translation of the output sentence. The OptPR algorithm is the method chosen for matching the source and target TSs, as it has

been shown to be very accurate for the purposes of string matching. In order to use this algorithm, three probability distributions must be acquired. In this case they are generated using the GIZA++ utility, an implementation of the IBM Models 1-5. In an attempt to improve accuracy, sentences are processed after being segmented into three phrases. With a unique model built for each phrase type, the string matching algorithm can be specialized with respect to the distinct features of their respective structures.

5.1.2 Purpose

The purpose of this approach is to explore the viability of using Syntactic PR methods in the field of MT. There are several statistical approaches used in SMT systems, but most operate using probability models to generate the output, rather than matching the input structure to a specific observed pattern. As a result, the output of an SMT system will have a certain probability of belonging to the TL, but there is no guarantee that the output sentence is actually valid without additional post-processing. The obvious reason why an SMT does not match against actual instances of sentences is that it is impossible for a parallel corpus to contain every potential sentence of the TL. If such a corpus did exist, the translation system would not be necessary. This system hopes to achieve the benefits of pattern matching without requiring an exhaustive list, by using the TSs described in Chapter 4.

5.1.3 Roadmap

The following section lists the various contributions that this thesis makes to the field of MT in regards to the usage of syntactic PR, as well as describing past research that these contributions build off of. Thereafter, the purpose of each individual contribution is given, as well as the details of the approach used.

5.2 Contributions of this Chapter

The major contributions of this thesis, in regards to the syntactic PR enhancements to MT, are listed in this chapter. We will first list the prior research related to the contribution, and follow it by the contribution itself and how it builds on the previous research. The contributions in this section are: The usage of word alignment utilities in a unique approach involving tags rather than words, the segmentation of sentences to apply phrase-based methods in our translation system, and the utilization of the OptPR algorithm in the domain of MT.

5.2.1 Contribution: Word Alignment Utilization

As discussed previously, a parallel corpus is a collection of sentence pairs that have the same meaning, but are written in different languages. When these are used to generate models that estimate the probability of a word in the SL becoming a word in the TL, this process is referred to as “Word Alignment”. GIZA++ is an implementation of various word alignment model generators, including the IBM Models 1-5 [27] , and is the most widely used word alignment utility. Various files are generated using GIZA++, including one that details the probabilities of a token in the SL turning into a token in the TL, based on the parallel corpus used. As the parallel corpus that this project uses consists of PoS tags rather than individual words, the vocabulary of the corpus is greatly reduced, and presents the GIZA++ utility with a set of data whose attributes it would not typically deal with. We now explore how the GIZA++ handles this departure from its standard usage, particularly regarding the results of the IBM models. The output of the GIZA++ utility is used in the creation of the models needed for the OptPR algorithm.

5.2.2 Contribution: Sub-Sentence and Language Division

SMT systems that rely on word alignment models often produce non-grammatical sentences. While using word alignment probabilities does produce a target sentence t that is highly likely to correspond to the source sentence s , the target sentence

is not actually chosen from a collection of sentences that is known to exist in the TL. This naturally results in word alignment translation over-generating, producing sentences that do not actually exist in the TL. A method of resolving this is to perform alignment at a level greater than individual words. Obviously, sentence-to-sentence alignment is impossible, as it would require a parallel corpus containing every possible sentence in either language, defeating the purpose of even needing a translation system. Instead, phrase-based methods attempt to perform alignment at the phrase-level, mapping multi-word phrases that commonly co-occur between the SL and TL. We borrow this concept here by performing phrase-level alignment using the word-alignment probability models. However, while standard phrase-based alignment generates a static list of phrase pairs, with a score of the likelihood of them co-occurring, this thesis instead makes use of a static list of TL phrases, with the SL phrases being dynamically extracted from the input text. The score is then assigned using the OptPR algorithm. These scores can then be used to evaluate the overall probability of the source TS matching the target TS.

In contrast to what is said above, it is also important to break down large structures to reduce the complexity of the pattern being processed. Translation systems have a tendency to perform well on small sentences, but degrade in quality as the size of the input sentence increases. To combat this, some approaches elect to break down the sentences into smaller components during the pre-processing phase, rather than performing translation one sentence at a time. One such translation system uses parse trees to divide input into sub-sentences, for the purpose of Chinese-English translation [40]. In this case, commas are used to split the parse tree into sub-structures. These structures are then validated to ensure that the sub-sentence has enough syntactic information for translation. The benefit of this approach is that it reduces the size of the segment being translated, but also introduces a number of issues. The most obvious problem is that the sub-sentences will have to be aligned themselves after translation. Additionally, it is possible that a proper translation would have components of one sub-sentence move to another. A form of this approach is used in this thesis, as the sentences are processed using three syntactic components of a sentence: the subject, object, and verb phrase. These were initially extracted to validate the

training data, but can be repurposed here.

In addition to reducing the complexity of the strings being processed, we hope to also make use of the benefits typically associated with a “domain language” (DL). A DL is a subset of a natural language used in a specific domain. A DL translation system should outperform a general purpose translation system, as the words used will have less potential meaning and the syntax tends to be more structured, especially in domains such as technical documentation. An example of an area in which the concept of a DL is applicable is the medical domain. A large amount of resources are available in this field, including the “Unified Medical Language System” (UMLS), which contains a semantic graph that maps a large amount of terms commonly used in the medical domain between languages. This resource has been used to augment an SMT system for Spanish-English translation in this domain [38]. By using the word mappings that the UMLS provides, the system was able to significantly out-perform a base SMT translation system trained in medical texts.

In this thesis, we work with sentences that fit the standard Subject-Verb-Object structure. As we are extracting each component and treating them as individual phrases, it could be useful to consider each phrase type as belonging to their own sublanguage, with their own frequency of word-types and syntax. Building models specifically trained for these phrase types could then reap the benefits that a DL system does, as the syntax used to construct the object phrase will differ from what is used in verb phrases.

5.2.3 Contribution: MT Using OptPR Algorithm

The OptPR Algorithm, covered previously in Chapter 3, is a Syntactic PR algorithm that performs string matching. In the paper published, it is shown that it performs highly accurate string matching even if the estimated model has an unreasonable amount of error. In regards to this project, the OptPR algorithm is used to match the input English phrases with Japanese phrases collected from the parallel corpus. In addition to the performance of OptPR, the translation system will also benefit from having the phrases matched up with actual instances found in the corpus. The

phrase templates retrieved are then assembled in the standard Subject-Object-Verb orientation of a Japanese sentence, constructing a Japanese TS that can then be used to construct the output text of an English-Japanese MT system.

5.3 Word Alignment Utilization

This section will cover how a portion of the OptPR algorithm's model is acquired through the usage of existing word-alignment tools, namely GIZA++. By adapting the output of the IBM Models, both the substitution and qualified insertion distributions can be acquired from the parallel corpus that has been previously transformed into a set of TS pairs.

5.3.1 Purpose of Contribution

The purpose of this contribution is to provide some estimation of the models required for the OptPR algorithm to function. Unlike the previous example given, where the algorithm was used to match noisy English words to a set amount of words in a dictionary, the noisy English TSs and the dictionary of Japanese TSs do not have the same “alphabet”, or tag set. While the English noise from the mistypes on a keyboard can be estimated using the proximity of keys to the correct key, there is no obvious way to easily estimate the probabilities of an English tag becoming a Japanese one. While the IBM Models were not developed for the purpose of string matching against a preset dictionary, its output can be used to obtain these estimates.

5.3.2 Description of Contribution

Word Alignment tools, specifically GIZA++, are used to produce the probability distributions that are required for the OptPR algorithm to function properly. The input that must be supplied are two files containing the raw sentences of the respective languages, with equivalent sentences of either language occurring at the same line number. The pre-processing stage generates co-occurrence files, counting the number of times two tokens of the respective language occurred within the same sentence.

Additionally, the number of times a token appears in the entire corpus is counted, so that one can calculate how often a token appears with another against how often it appears by itself.

In this case, the tokens are members of the tag set. As the name implies, word-alignment tools are typically used for words rather than generic placeholders. Two major differences between working with words versus tags are:

1. There much fewer tags than words, and so they will appear in the corpus far more frequently
2. Tags have no semantic meaning.

The first difference means that there is a generalization of the data, implying that significant alignment information could be lost. On the other hand, since the data has been generalized to purely syntactic representation, the alignment of the syntax of the two languages can be approximated with far less samples. The second difference should only be an issue if the semantics of the word components influence the structure of the sentence itself during translation. As this thesis is concerned with structural translation, the training data has been filtered to prevent the inclusion of such semantic translations, but it is possible that there are semantic translations left in the data that abide by the structural filters. However, to prevent the working training set from being reduced to an unusable size, we did not resort to any further filtering.

The GIZA++ executable itself runs the data against a number of model generation algorithms, including the IBM Models 1-5. The result of this is a large number of output files, which must be interpreted in a way specific to the Opt PR algorithm's mechanics. Of particular interest is the file with the extension of ".t3"[36]. An excerpt of this file is shown below.

Figure 5.1: The t3 file contains a probability distribution for the one-way mapping between the SL and TL, based on the parallel corpus provided. As the OptPR algorithm operates by assuming the input is a noisy version of a dictionary entry, the Japanese language is considered the SL in order to obtain the proper distributions.

The main t3 file contains three columns. The first two map to IDs for tokens in the SL and TL respectively. These IDs reference the actual tokens in the language’s vocabulary file. The third column is the probability of the SL token becoming the TL token during translation. This output can be used to generate two of the distributions needed for the OptPR algorithm, specifically the qualified insertion and substitution distributions, although some alterations must be made.

What is of importance here is how GIZA++ expects the models to be used, and how the OptPR algorithm actually operates. The models generated by GIZA++ are based on the theory that the source sentence has a probability of translating into every possible sentence in the TL. The translation algorithm then generates a sentence that belongs to the TL based on the translation and language models. The essential flow of this process is the sentence in the SL transforming into a sentence in the TL. The OptPR algorithm works contrariwise, viewing the input text as a result of various permutations performed on one of the entries in its dictionary. Due to this, in terms

of standard word-alignment procedure, the TL and SL are flipped for the purpose of generating a distribution that the algorithm can use. The t3 excerpt above shows an alignment of Japanese (SL) tags to English (TL) tags, which is what we need for the confusion matrix. Another issue is the way in which the “Null” value is mapped using GIZA++. The probability of an SL word becoming a “Null” tag is not calculated, but instead the probability of a “Null” tag becoming a TL word is. In terms of the OptPR algorithm, the confusion matrix that only includes the t3 values from the Japanese-to-English word alignment would not include the deletion component (the probability of a Japanese tag being substituted for “Null”). This is, instead, acquired by re-running the word alignment as English-to-Japanese. The probability of a “Null” tag becoming a TL word (in this run, the TL is Japanese) can be interpreted as a TL word being substituted for a “Null” tag. The two images below show the word-alignment probability distributions that can be obtained, and how the “Null” values are mapped depending on the direction in which the word-alignment is run.

	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	Null	DT	NN	NNS	JJS	PRP	JJ	POS	PRPS	NNP	IN	TO	CD	PDT	CC	RB	''	'	VBG	.	VCN
0 Null		0.69584	0.0001	0	0	0	0	0.0078	0.0864	0.024	0.14	0.008	0.025	0.003	0.003	0.001	0	0	0	0.0001	0
2 NN	0	0.0001	0.8006	0.154	0	0.0312	0	0	0	0.013	0	0	0.0001	0	0	0	0	0	0	0	0
3 を	0	0.53	0	0	0	0.039	0	0.003	0.3975	0.016	0	0	0.006	0.0001	0	0	0	0	0	0	0
4 RB	0	0	0.006	0.188	0.183	0	0	0	0.072	0	0	0	0	0	0	0.549	0	0	0	0	0
5 JJ	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6 の	0	0	0	0	0	0	0	0.153	0.0001	0	0.8405	0.005	0	0	0	0	0	0	0	0	0
7 に	0	0	0	0	0	0	0	0	0.0001	0	0.8602	0.139	0	0	0	0	0	0	0	0	0
8 RP	0	0	0.0796	0	0	0.458	0	0	0.462	0	0	0	0	0	0	0	0	0	0	0	0
9 NNP	0	0	0	0	0	0	0	0	0	0.904	0	0.001	0	0	0	0	0.047	0.047	0	0	0
10 へ	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
11 VB	0	0.046	0.562	0	0	0.0001	0	0	0	0	0	0	0.13	0	0	0	0	0	0.1304	0	0.1301
12 CD	0	0	0	0.096	0	0	0	0	0	0.0001	0	0	0.902	0	0.0006	0	0	0	0	0	0
13 で	0	0	0	0	0	0	0	0.0001	0	0	0.999	0	0	0	0	0	0	0	0	0	0
14 と	0	0	0	0	0	0	0	0.0707	0	0	0	0	0	0	0.929	0	0	0	0	0	0
15 から	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
16 た	0	0	0	0	0	0	0	0	0	0	0.616	0	0	0	0	0	0	0	0	0	0.383
17 でも	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 5.2: The probability distribution obtained from running the word alignment tool with the Japanese language as the SL and the English language as the TL. The non-null Japanese tag rows are used for the substitution distribution, and the null Japanese tag row is used for the insertion distribution.

	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	Null	NN	を	RB	JJ	の	に	RP	NNP	へ	VB	CD	で	と	から	た	でも
0 Null	0	0.98	0.0001	0.001	0.002	0.001	0.0001	0.001	0	0	0.01	0.0001	0	0	0	0.0001	0
2 DT	0	0.0001	0.9999	0	0	0	0.0001	0	0	0	0	0		0	0	0	0
3 NN	0	0.492	0.333	0.02	0.0001	0.1104	0.011	0.0072	0	0	0.004	0.0167	0	0.0001	0	0.002	0
4 NNS	0	0.097	0.823	0	0	0.0001	0	0	0	0	0	0.0572	0	0.021	0	0	0
5 JJS	0	0	0	0.503	0.496	0	0	0	0	0	0	0	0	0	0	0	0
6 PRP	0	0.103	0.722	0	0	0.126	0.047	0	0	0	0	0	0	0	0	0	0
7 JJ	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
8 POS	0	0.0001	0	0	0	0.999	0	0	0	0	0	0	0	0	0	0	0
9 PRPS	0	0.0001	0.98036	0.0001	0	0.0001	0	0	0	0	0.005	0	0	0.013	0	0	0
10 NNP	0	0.0001	0.459	0	0	0.021	0.016	0	0.501	0	0	0	0	0	0	0	0
11 IN	0	0.0001	0	0	0	0.574	0.186	0.004	0	0	0.004	0	0.156	0	0.064	0.009	0
12 TO	0	0.0001	0	0	0	0.005	0.497	0	0	0.25	0	0.121	0	0	0	0	0.125
13 CD	0	0	0.13	0	0	0.12	0	0	0	0	0	0.738	0	0	0	0	0
14 PDT	0	0.003	0.019	0	0	0	0	0	0	0	0.977	0	0	0	0	0	0
15 CC	0	0	0	0	0.132	0	0	0	0	0	0	0	0	0.867	0	0	0
16 RB	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
17 ``	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
18 '	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
19 VBG	0	0	0	0	0	0.0008	0	0	0	0	0.999	0	0	0	0	0	0
20 .	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21 VBN	0	0	0	0	0	0	0	0	0	0	0.666	0	0	0	0	0.333	0

Figure 5.3: The probability distribution obtained from running the word alignment tool with the Japanese language as the SL and the English language as the TL. The null English row is used for the deletion distribution.

By combining the word alignment outputs of both Japanese-to-English and English-to-Japanese, the full confusion matrix can be obtained. The quality insertion distribution can also be acquired through the Japanese-to-English t3 file, as the Null-to-English-Tag probabilities are equivalent to the insertion probability of an English tag.

5.4 Sub-Language Models

This section will cover how the concept of sub-language processing and phrase-based methods are implemented in an effort to improve the accuracy of the string matching process. The sentences in the parallel corpus have been structurally divided into three different phrase-types, as described in Chapter 4. The benefits and reasoning of this,

as pertaining to the Syntactic PR are now discussed here.

5.4.1 Purpose of Contribution

The purpose of breaking down the string matching operation to handle three separate phrase-types is to leverage the benefit of having probability distributions fine-tuned for a specific problem, rather than a general model. As the phrase-types tend to have different structures in both the SL and TL, the string matching can become more accurate by performing on a sub-sentence level, and then forming the sentence procedurally afterwards. This also removes the problem of significant word re-ordering between the two languages, caused by the S-V-O and S-O-V base sentence structures.

5.4.2 Description of Contribution

The actual process of dividing the base sentences within the parallel corpus into the three component phrase-types is described in Chapter 4, as part of the effort to filter out semantically-driven sentence translations. While the benefit described in Chapter 4 relates to the heightened likelihood of the training data consisting of syntax-based translations, there are additional reasons for dividing the parallel corpus in this manner. The first benefit is that of leveraging phrase-based methods in assigning a score to a phrase-pair. As, by nature, the OptPR algorithm returns a probability for an input string Y being the result of a dictionary string U after passing through the noisy channel, a tuple can be dynamically generated that assigns a score to the phrase pairs. This scoring can then be used to help improve the string matching accuracy, described further in the OptPR section. The image below shows an example of a tuple that would be generated dynamically.

Figure 5.4: Using the OptPRs probability score, the input phrase can be ranked against the phrases contained in the algorithm’s dictionary, which can then be used to calculate the most probable sentence structure.

Additionally, this approach reduces the problem of arbitrary sentence length. By breaking down sentences into a set of phrases of which a sentence can be composed of, the translation system is less likely to experience a decline in performance based on the length of the sentence. There are a number of issues with this approach, including how these phrases are extracted. In our case, we choose to extract the so-called “Subject”, “Verb”, and “Object” phrases from sentences in the parallel corpus in a procedural manner, using the parse tree and particles of the English and Japanese sentences, respectively. The trade-off of this approach is the exclusion of more complex sentence structures, in favour of having a dataset that is more likely to consist of syntactic translations. Having more complex extraction methods to account for different components of complex sentences would be needed for complete language coverage, but the reduced scope will also allow for the OptPR algorithm to be evaluated using a clean training and test set. This system can then be extended to cover larger portions of the respective languages. Even when the scope is expanded, the system will still be able to mitigate the effects of sentence length, as it will simply mean additional sub-sentence models being constructed.

5.5 MT Using OptPR Algorithm

This section will cover the actual process of how the OptPR algorithm is used in a system meant for MT. As mentioned, this algorithm has been designed so as to be used to perform string matching, based on the three probability distributions acquired regarding the three phrase-types. The individual matching of the three phrase types

will be discussed, as well as the recombination of these phrases and the post-processing stage to improve the overall accuracy of the system.

5.5.1 Purpose of Contribution

The purpose of this contribution is obviously to fulfill the evaluation of the feasibility in using syntactic PR for MT. All of the steps taken up to this point were in the interest of creating a model best suited for the OptPR algorithm. The goal of this present contribution is then to utilize what has been constructed in an effective way, in order to yield a result that shows the viability of syntactic PR methods.

5.5.2 Description of Contribution

As the reader understands, the main component of this thesis is the usage of syntactic PR algorithms to assist in MT, specifically the OptPR algorithm. This algorithm has been used to perform string matching between garbled and ungarbled English words, and we hope to utilize essentially the same functionality for a different purpose. Along with the three probability distributions covered earlier, the algorithm contains a dictionary and an alphabet. The dictionary is the set of strings that the input is matched against, and the alphabet contains the elements that compose both the strings within the dictionary and the input strings.

In regards to OptPR being used for string matching English words, the dictionary consists of the ungarbled words, and the alphabet is obviously the English alphabet. In this case, both the input strings and dictionary words are composed of the same elements, and so use the same alphabet. Because this is the case, it can be expected that the substitution model will heavily weight the substitution of an element in the dictionary word for the same element in the noisy word.

For this MT project, the substitution model will not be as straightforward. The input strings are English TSs and the dictionary words are Japanese TSs. While the TSs of the respective languages have some common tags, there are also a large number of tags that are exclusive to each language. This results in two distinct alphabets for the dictionary and input strings.

As discussed, the OptPR algorithm will operate using three distinct models for the “object”, “verb”, and “subject” phrases extracted from the corpus. As such, the three dictionaries corresponding to each phrase type consist of the TS segments for these phrases.

The OptPR algorithm is then used to match an input phrase TS against all of the TS phrases in the dictionary. The hope is that there is enough consistency with the sub-structure of these phrases that meaningful PR can be performed. The downside here is that, as these sub-structures are obviously smaller than the full structure of the sentence, and the alphabet that composes these sub-structures is finite, there are considerably less unique strings representing these phrases than that of a whole sentence. Because they are less unique, it is more likely that the same SL input string will correspond to two different TL dictionary entries in the test data. Since the OptPR algorithm is deterministic, this input string will at best correctly choose only one of these two dictionary entries, leading to an unavoidable amount of error. Additionally, as the three phrase types are matched separately, there is no guarantee that when combined the final TS will have existed in the corpus.

These factors are mitigated by an additional validation step, the purpose of which is both to improve the accuracy of the string matching, and to better facilitate the actual word-level translation. As mentioned previously, the same input string could match up with two distinct strings in the dictionary. Allowing for more than one string to be chosen by the OptPR algorithm will make it more likely that the correct string is included in the list of strings chosen. The extreme of this is allowing all strings in the dictionary as a possible match. Returning every string in the dictionary itself would be meaningless, but combined with the probability assigned by the OptPR algorithm, a ranked list can be formed.

From the three lists obtained for each phrase-type, the most probable TS can be obtained using the joint probability of each individual phrase. Obviously, the most probable combination would be the top phrases on each ranked list, but the combination of these three phrases may not form a TS that exists in the corpus used. Instead, the most probable combination *that exists in the corpus* is chosen, both to allow for different phrase structures to be chosen based on the same input

string, and to mitigate the possibility of composing a nonsensical sentence structure. Additionally, it must be ensured that the TS can actually be used for translation. The tags in a TS are ultimately placeholders for word-level translation to be performed, and so there needs to be a certain correspondence between the source TS and the target TS, particularly in terms of content words. For instance, if the object phrase of the input TS contains a noun, while the object phrase of the matched TS contains a noun and a verb, it is much more likely that the noun maps to the noun. The verb is then a “dead” placeholder, with no potential content to fill it. To prevent this from occurring, the system can also choose the phrase combination that both requires the full TS to exist in the corpus, and that there is a 1:1 alignment of content words. The tradeoff here is that it would aid in translation, but potentially choose the incorrect TS based on the parallel corpus.

5.6 Conclusion

The goal of this system is to facilitate English-to-Japanese translation, with a focus on increasing the likelihood of grammatically correct output. The differences between the two languages have been taken into account, motivating the usage of phrase-level methods and string-based representations. By generating a word alignment model using the TSs rather than raw sentences, we have placed an emphasis on the relationship of word classes and particles between the two languages, leading to a system that is more sensitive to syntactic similarities, rather than a standard SMT.

We have also argued that the OptPR algorithm, proposed by Dr. John Oommen [28], is well-suited for this approach, more so than standard SMT systems. The use of a preset dictionary, which would normally be a major limiting factor for translation, is able to cover a significant portion of the target language through the use of TSs. This allows for the translation system to select a sentence *structure* that has appeared in the training data, without requiring a prohibitively large set of sentence pairs to make this approach feasible.

Chapter 6

Experimental Setup and Results

6.1 Introduction

6.1.1 Chapter Overview

In this chapter, we will describe the various experiments used to evaluate the concepts proposed in this thesis, as well as present and analyze the subsequent results. The purpose of this chapter is to show that the ideas presented here have been implemented and tested to assess the strengths and weaknesses of not only the system as a whole, but of each component as well. In addition to creating our own implementation, a baseline SMT system has been trained using the TS representations. The results acquired through this SMT system can then inform us as to whether the results of our own approach are in-line or superior to existing architectures in working with the TS data.

6.1.2 Road Map

In the following section, the process of acquiring and factoring the data set needed to both train and test our system will be given. Afterwards, the various experimental setups used to evaluate the system in different ways are explained, as well as the reasoning behind each setup. Following that, the construction of the baseline SMT system will be briefly covered, including the various packages used and the results

obtained. Finally, our experimental results will be presented along with analysis of each set-up individually, as well as an explanation of how the results of our system compare to those of the baseline SMT.

6.2 Data Acquisition and Processing

In this section, we will briefly cover the corpus used in our experimentation, including how it has been filtered to reach a state that is both usable and beneficial for our system. As the focus of our system is syntactic translation, it is important that the training data used contains sentence pairs that are structural translations, rather than semantic. By filtering the data in specific ways, we hope to automatically extract such translations, while maintaining enough data in the corpus so that a meaningful model is trained.

The corpus used in our experimentation is the Tanaka corpus. As part of an assignment, Professor Tanaka of Hyogo University had his students gather English-Japanese sentence pairs, compiling what they had collected into a parallel corpus[34]. Through several years of collecting these sentence pairs, the corpus grew to be a potentially useful resource for MT. Since the pairs were provided by students, the primary source for these sentences were traced back to text books designed to help Japanese students learn the English language. Beyond that, the other two major sources were song lyrics and translated fictional novels.

Initially, there were a great deal of errors and repeat examples contained within the corpus, but the collection has since been revised and corrected, reducing the number of sentence pairs to just over 150,000. It is this revised set that has been used in the experiments for this thesis, currently maintained by the Tatoeba group [10]. While the corpus has gone through a great deal of revisions since the initial collection, due to the nature of the sources and how they were gathered, multiple issues still exist when working with the Tanaka corpus. As the sentences were inputted by students, there still exist several errors throughout the corpus. Additionally, a large number of both the English and Japanese sentences are either antiquated or syntactically incorrect.

In order to help ensure that the data used during experimentation is valid, additional filtering steps are used to validate the sentence pairs, at the expense of reducing the size of the data set. The two filters used are syntactic filtering (filtering out sentences that don't have the same phrase types) and content agreement filtering (filtering out sentences that do not have the same number of content words). These filtering processes have been covered in-depth previously in Chapter 4.

After applying the syntactic filter, the number of usable samples is reduced to 2807. This large reduction can be attributed both to the fact that the corpus contains a large number of semantic translations, as well as the restrictive behaviour of the filter. As the filter assumes a standard S-V-O structure of English sentences (or S-O-V for Japanese sentences), certain more complex or longer sentences may be excluded. The benefit of working in this reduced scope is that there is a much higher guarantee that the training data will be useful in building a syntactic translation system. The downside is that the system will not be reflective of an entire language, but the results attained on this subset can potentially justify expanding the system to allow for more complex sentence structures.

After applying the content agreement filter, the set of data is further reduced to 834 sentence pairs. While a standard SMT would need a corpus that is orders of magnitude larger than this, the benefit of working with TSs is that a much larger window of a language is covered with a small set of sample data.

Based on these filters, two data sets are used in our experiments: one with merely syntactic filtering applied containing 2807 sentence pairs, and the other with both syntactic and content agreement filtering applied containing 834 sentence pairs. The set containing 834 instances will be used as the training data for our primary evaluations, but the less restricted set will also be analyzed to capture how well the system performs with potentially noisier data.

6.2.1 Grammar Used in Experimentation

As mentioned earlier in Section 1.4.1, our current approach works within a limited scope of the complete English and Japanese languages, placing restrictions on the

grammar that this system can process. On the English side, we limit the acceptable sentences to be of the standard subject-verb-object structure. To achieve this, we first obtain a constituent parse of the sentence being processed, obtained using the OpenNLP toolkit[1]. We then ensure that the three child nodes of the sentence node are (in the specified sequence) “NP” (Noun Phrase), “VP” (Verb Phrase), and a period. Thereafter, we check that the VP discovered has two children, consisting of a verb (or another VP) and a NP child. As for the Japanese language, we verify that the sentence consists of a verb, as well as subject and object markers. As such, our current experiments do not deal with certain major issues that affect MT, such as structural ambiguity[14], which we suggest as a future research avenue.

6.3 Experimental Configurations

In this section, the experimental setups used to evaluate our approach will be detailed. Three major setups have been used to test the concepts in this thesis, in order to gather enough data to analyze the strengths and weaknesses of the system. Additionally, each setup contains three phases of evaluation that correspond to the input TSs progression through the system. In Chapter 5, a flow of events was given in terms of how an input sentence would be matched against a target TS. Each of these steps are evaluated separately in their respective phases, both to see how well the system performs at a base level, as well as to see how performance is impacted at later stages in the matching process. Additionally, the concept of “Tag Simplification”, described in Chapter 4, is also evaluated in its own setup.

The input of each phase is a test set of source (English) TSs, each of which is matched to a target (Japanese) TS through varying levels (phases) of string matching. The target TS obtained through the system is then compared against the expected target TS as gathered from the original sentence pair. The metric used in evaluating the performance of the string match is the Levehnstein distance between the actual and expected result, with each TS considered a string and each tag corresponding to a single character. The data sets used are partitioned into a 70/30 split for training and test data.

Each individual experimental setup will be described below, as well as the three phases of evaluation.

6.3.1 Setup-1: System with Primary Data Set

Setup-1 is the main experiment of this thesis and uses the fully filtered data-set to train and test the system. By evaluating this setup we will be able to gauge the performance of the various concepts presented in this thesis, as well as the viability of using this approach as a whole.

6.3.2 Setup-2: System with Syntactic-Filtered Data Set

This setup is similar to Setup-1, except that the data set used is less filtered, meaning that the model will potentially cover a greater area of the TL at the cost of using data that will likely contain more instances of sentence pairs that are not structurally equivalent. This setup is primarily to evaluate the three phases with a less-than-ideal data-set.

6.3.3 Setup-3: System with Tag Simplification

This setup allows for frequently co-occurring tags in a TS to be grouped together, using a method called “Tag Simplification” covered in Chapter 4. By combining tags that frequently appear together, a stronger correlation between tag sequences may be discovered, at the cost of potentially over-simplifying the TS. The decision to experiment with and without tag simplification is due to it having a potentially negative impact on the performance of the system. In particular, it is expected to reduce the accuracy of approaches involving separating the original TS into smaller phrases (Phases 2 and 3). As the phrases are already the result of segmentation, further reduction in their lengths would, very likely, result in over-simplification. The reason for including this option then is not only to see if the expectation of reduced performance is correct, but to also test if it improves the system’s performance when attempting string matching at the complete TS level. If it does improve the

performance, it can then be argued that tag simplification would also be beneficial when phrase lengths are sufficiently large. The data-set used for this setup is the fully filtered one, as the primary interest of this setup is the first phase, which is the one least influenced by the data-set chosen.

6.3.4 Phase-1: Base String Matching

The first phase performs string matching on the source TS without the use of any phrase-based methods. The input TS is directly string-matched to a target TS using the OptPR algorithm, and the target TS is then compared against the expected result. The purpose of this phase is to evaluate how well the system performs without any assistance. The test TSs are being run through the OptPR algorithm and being matched to TSs in the target language, with only the probabilities generated using the training data to assist in the process. This stage is thus performing the syntactic PR portion of the system in its purest form, and its results will help inform us about how viable this approach of Syntactic PR is in the domain of MT. The results obtained here can also be compared to later phases to analyze the accuracy improvements that each additional processing stage provides.

6.3.5 Phase-2: Content Agreement

The second phase performs string matching with content word agreement. As the basis of this translation system is to create a mapping between the syntax of two languages, the target TS ideally has the same content word tags so that those slots can be filled. Due to the way that the corpus was filtered to help ensure that the sentence pairs are syntactic translations, the results obtained for this phase may not accurately portray the accuracy boost given by this stage. To try and isolate the performance increase gained through this phase, the results of the secondary experimental setup are important for correctly analyzing this phase. The issue with doing so is that the data set used will be less reliable, but the results obtained should still be somewhat indicative.

6.3.6 Phase-3: Dictionary Matching

The final phase combines the three phrase types into the full TS, with the condition that the acquired target TS exists in the training data. To do so, the OptPR algorithm assigns a probability value to each potential target TS for each of the three phrases, creating a ranked list of every potential target TS phrase. The highest combined probability of the three phrase types that exists in the training data is then used as the final result. Since Phase-2 was evaluated using two data sets, it is important to also analyze the secondary setup to ensure that the performance improves without the benefit of content agreement filtering. The results obtained using the fully filtered data set will be more heavily weighted during analysis, as it is trained on data that is more likely to be syntactic translations.

6.4 Baseline Experiment

This section will cover the SMT system specifically trained to compare against the results of our system. In order to properly evaluate the power of the system proposed by this thesis, it is important to have results of existing translation systems against which it can be compared. Due to the unique representation of TSs used in this thesis, no existing data is available that our results can be directly compared against. Instead, we have opted to construct an SMT using standard tools with the same data used in our own experiments. Three major packages have been used in constructing the SMT used to create the baseline results, briefly covered below.

6.4.1 Moses Statistical Machine Translation System

Moses is a widely-used set of tools that allows for the generation and testing of a complete SMT system. It is capable of generating a translation model that uses the output of a word-alignment tool (such as GIZA++), and further processes the information to create a phrase-based translation model. This translation model can be loaded alongside a language model of the TL to perform translation on input given by the user.

6.4.2 IRST Language Modeling (IRSTLM)

The IRSTLM tool[11] creates a language model from a collection of tokenized sentences, and outputs the model in a form that is usable by statistical translation software. The toolkit allows for various configurations when constructing the model, and for the purposes of this experiment we use a 3-gram model with Kneser-Ney smoothing, as advised by the instructions provided by the Moses SMT system.

6.4.3 GIZA++ Word Alignment Tool

As a tool previously used in our own experimentation, it is suitable to use GIZA++ to help construct the baseline SMT as well. While the usage of GIZA++ in terms of configuration may be somewhat different when used as a component of Moses' own model generation, the same training data used to train our main experimental model is used here, as is the distortion table with which we attained the confusion matrix.

6.4.4 Baseline Results

The baseline models were created using the same 70/30 split for obtaining the training and testing data as used in our system experiments, using the fully-filtered data set. By feeding the system the English TSs contained within our test data, Japanese TS translations generated by the SMT were obtained. The Levenshtein distance between the actual and expected result is used to evaluate each translation, with each tag in the TSs being considered as a single character. The results of the experiment are shown in the table below.

The individual counts of each distance value will be important in comparing against our system's experimental results, as it not only captures the distribution of distances attained using a particular method, but also shows how many perfect matches were made.

Additionally, the average distance of the translations is 3.93, with an average length of 8.46.

Table 6.1: The results of the Baseline experiment using the Levenshtein distance as the evaluation metric. The “Distance” row shows the distance value attained when comparing the actual and expected result, and the “Count” row shows the number of instances in the test data that evaluated to that distance.

Distance	0	1	2	3	4	5	6	7	8	9	10
Count	0	10	59	47	45	35	27	17	7	1	1

6.5 Experimental Results and Analysis

In this section, the results of each experimental setup will be provided, along with an analysis regarding both the evaluations of each individual phase as well as how the results evolve throughout the phases. Afterwards, the primary experimental setup is compared against the results of the baseline system, showing the strengths and weaknesses of our approach in comparison to standard SMT methods. Finally, an overall evaluation of the system will be given.

The following sections will present the results of each phase in a manner similar to the results of the baseline SMT. A table will show the number of tests that attained varying degrees of distance, followed by a discussion of what can be learned from the results.

6.5.1 Setup-1

This section will cover the results of Setup-1, and how each of the phases compare with each other. The table below shows the distance counts of each phase, followed by a graph that represents the results of each phase as an individual line. The average length of the expected Target TSs is 8.34 tags.

Table 6.2: Results of the various phases for the experimental Setup-1. A clear improvement between each phase can be seen, with a large number of perfect matches being attained by Phase-3.

Distance	0	1	2	3	4	5	6	7	8	9	10
Phase-1 Count	0	1	37	40	73	63	26	5	0	1	0
Phase-2 Count	7	13	59	66	41	25	17	6	2	5	5
Phase-3 Count	13	51	49	50	41	22	8	7	4	1	0

Figure 6.1: Line Graph of the results for Setup-1. The peaks of each phase can be seen to migrate closer to the left side, indicating an increase in the overall accuracy.

As can be seen, each phase has an increase in performance when compared to the previous one. Phase-1 has a large peak in the middle values, and a sharp decline on its right-side tail, dropping below the other two phases at distance 7. This shows that the base OptPR algorithm is able to use the probability models created from the training data to fairly accurately match against the test data, but fails to fully “zero” in on an exact match for the input TS. This is most likely due to the Japanese-specific tags having no direct correlation in an English tagset. With the confusion

matrix being built on whole sentences, tags that frequently occur in one section of a TS would cause the model to believe that the tag occurs frequently in other parts of a TS as well. Lacking segmentation into models built on each phrase-type, the OptPR algorithm is capable of matching TSs that are close to the actual result, but fails to achieve a perfect match.

Phase-2 achieves 7 perfect matches, and generally performs better than Phase-1. The issues associated with Phase-1 are alleviated here through the use of models trained on each phrase-type, with the peak of phase 2 moving closer to the left-side of the graph. The downside of this phase is that it combines the three phrase-types without regard for the issue whether the resultant TS exists in the training data or not. The high counts in the 7-10 range are likely caused by this phenomenon, as the resultant combination has potential to be syntactically incorrect, and so, by default, should fall some distance away from the expected TS. In the very worst scenario, it will create a TS that has a greater distance than any that could be chosen by Phase-1.

Phase-3 achieves 13 perfect matches, and has fairly consistent counts in the 1-4 range. This shows that even if the system is not able to perfectly match to the correct target TS, it is still likely to be very similarly structured. This phase essentially takes a combination of the phrase TSs attained in Phase-2 that exist in the training data, so that the results are once again guaranteed to be based on a sentence structure observed in the TL. The additional boost in performance is likely due to the removal of nonsensical results produced by Phase-2.

6.5.2 Setup-2

This section will cover the results of Setup-2, which is essentially an experiment to see if phases 2 and 3 will still achieve improvements in performance without the content agreement filtering performed on the corpus. Consequently, the training data is more likely to be erroneous. The table below shows the distance counts of each phase, followed by a graph that represents the results of each phase as an individual line. The average length of the expected Target TSs is 9.53 tags.

Table 6.3: Results of the various phases for the experimental Setup-2. While we do not obtain as many perfect matches as in Setup-1, there is still an increase of performance at each phase.

Distance	0	1	2	3	4	5	6	7	8	9	10	11	12
Phase-1 Count	0	11	25	109	150	234	155	76	51	18	7	3	1
Phase-2 Count	6	15	79	127	200	148	94	53	42	45	21	10	0
Phase-3 Count	9	67	102	183	174	123	74	59	31	10	6	2	0

Figure 6.2: Line Graph of the results for Setup-2.

The Phase-1 results have a similar shape to its counterpart in Setup-1, except that it has sharper declines on either side. Additionally, there are improvements with each succeeding phase, though slightly less pronounced. In particular, the ratio of perfect matches to the total number of test samples is greatly reduced. It is likely then that the content-matched corpus allowed for more perfect matches to be attained, but the additional phases are capable of improving the performance all the same. The right-side tail of Phase-2 is also more pronounced, as the longer strings result in a great deal more distance between the expected results and nonsensical TSs that Phase-2 can achieve.

6.5.3 Setup-3

This section will present the results of performing tag simplification and dissect why this approach failed. The table below shows the distance counts of each phase, followed by a graph that represents the results of each phase as an individual line.

Table 6.4: Results of the various phases for the experimental Setup-3. Phase-2 and-3 attain the same results, which is likely the worst distances possible given the test data.

Distance	0	1	2	3	4	5
Phase-1 Count	2	157	17	3	0	1
Phase-2 Count	0	0	1	174	4	1
Phase-3 Count	0	0	1	174	4	1

Figure 6.3: Line Graph of the results for Setup-3.

Firstly, it should be noted that Phase-2 and 3 yield equivalent results, and this is likely because the distance counts are the maximum allowable for the TSs being considered. This result is unsurprising, as these phases employ phrase-based methods and so were already sub-sentences before they were simplified.

As for the results of Phase-1, it became evident that by combining frequent tag sequences, any error in string matching would result in a greater loss in information than if the tags were kept separate. When performing string matching without simplification, if the algorithm chooses a TS that is a distance of unity away from the expected result then there is only one incorrect tag. With simplification, however, a distance of unity could mean that there is any number of incorrect tags, depending on the tag combinations.

The results presented here indicate that tag simplification not only renders later stages of processing performed by our system useless, but also performs poorly on its own. Thus, we conclude that our string matching should be performed without this augmentation. Additionally, future efforts would most likely not benefit from this concept.

6.5.4 Baseline Comparison

This section will compare and analyze the results of Setup-1 and the baseline SMT system. As the baseline results were obtained using sophisticated model generation explicitly designed for translation, achieving comparable results should indicate the viability of this approach. Each phase of Setup-1 will be compared with the baseline, followed by an overall remark regarding the comparison.

The graph below shows the results of Phase-1 as compared to the baseline SMT.

Figure 6.4: Comparison graph between the baseline SMT results and the results of Phase-1. Baseline results are slightly superior, but they are comparable.

As can be seen, the baseline peaks earlier but has a less steep decline, resulting in it having higher counts in the 7-10 range. As mentioned previously, due to the models of Phase-1 being trained on complete sentences rather than having separate models for the phrase types, it is very capable of achieving a close match to the expected TS, but fails to achieve any perfect matches. The baseline is built on more sophisticated models, but still fails to achieve any perfect matches. The average distance achieved by Phase-1 is 4.02, which when compared to the baseline’s average of 3.93, shows that the results of the two are not greatly different. Furthermore, upon manually inspecting the target TSs attained by the baseline SMT, a large amount of the TSs were very close to the expected result, but were also syntactically invalid. As the OptPR approach has the additional benefit of guaranteeing that the TS chosen exists in the TL, our system has an edge in that it is more capable of allowing for syntactically-valid sentences.

Figure 6.5: Comparison graph between the baseline SMT results and the results of Phase-2. The results of Phase-2 show our scheme to be slightly superior to that of the baseline.

The results of Phase-2 are similar in shape to that of the baseline's, but a higher peak at distance count of 3 and a steeper decline on the left-hand side of the graph indicate that the performance of Phase-2 here is slightly superior to that of the baseline's. While Phase-1 can be said to be superior to the baseline due to it having more syntactically-valid results, the same cannot be said for Phase-2. Due to it not matching the complete TS against the training data, Phase-2 also suffers from producing syntactically-incorrect TSs.

Figure 6.6: Comparison graph between the baseline SMT results and the results of Phase-3. While the results of Phase-1 show that the OptPR algorithm can produce comparable results to the baseline with the benefit of syntactic correctness, Phase-3 shows that with further processing our system is capable of surpassing the baseline results while still maintaining that correctness.

The results of Phase-3 are greatly superior to that of the baseline’s, achieving an average distance of 2.85 as compared to the baseline’s average of 3.93. In addition, Phase-3 has the same benefit of Phase-1 as the target TS obtained is guaranteed to be in the TL.

6.5.5 Overall Evaluation

Overall, our system outperforms the baseline SMT in terms of both producing sentences which are syntactically-correct, and to achieve string matching with the correct TS. One factor that could contribute to this is that a standard SMT is designed to work with large amounts of data to create sophisticated models, and so working with a small data set is not exactly the intended usage. This works towards the concept of TSs in this thesis however, as our system is able to make better use of the small amount of data available in our data-set. Additionally, the same word-alignment models were used in the baseline SMT and our system, ensuring at the very least that the basis of the two models are very similar. While some of the accuracy of our final solution can be attributed to the content word filtering, Setup-2 shows that

these phases still achieve improved accuracy even without such a filter.

Finally, a major drawback of SMTs is their ability to create sentences that are close to ones that would appear in the TL but oftentimes have minor grammatical errors. With the usage of the OptPR algorithm this is much less likely to happen, as the output must have originated from a preset dictionary where the dictionary in this case is the target TSs in the training data. The reason for an SMT generating syntactically-incorrect sentences is due to the fact that they have to rely on a language model to approximate the probabilities of a sentence existing in the language. This model is necessary as it obviously isn't possible to store every potential sentence in a language for reference. Our system does not make use of such a language model, as the OptPR algorithm instead compares the input against a set of data. If we were working with raw sentences this would be the same issue that requires SMTs to use language models, but by rather making use of the TS form, while not spanning a complete language, we can create a sizeable coverage of the language with the data that is available. This allows our system to operate using the OptPR algorithm, which, in turn, allows us to guarantee that the syntax structure has appeared in the target language.

6.5.6 Standard Evaluation Metrics

During the course of the study, we did not resort to utilizing many of the standard evaluation metrics used in the literature. However, to place this thesis in the right perspective, we list a few that are commonly used in the field of MT.

Human evaluation is the ideal strategy when it comes to analyzing the quality of translations. This is, however, the most time-consuming option. As translations are meant to be used by humans, it follows that the output of an MT system should strive to meet human expectations. Additionally, any metric that automatically evaluates the quality of a translation should come as close as possible to matching human evaluation. A number of metrics have been defined for this purpose, with the BLEU metric (Bilingual Evaluation Understudy) being the most commonly-adopted one.

The BLEU metric operates by comparing a machine-translated “candidate” sentence to a number of human “reference” translations of the same sentence. Specifically, it observes the number of shared words and phrases between the candidate and the references, accomplished by counting the number of shared n -grams. Without restriction, this could lead to a candidate translation that simply repeats a single word contained in a reference, being scored very highly. To prevent this, a word or phrase in the candidate translation can only be counted for as many times as it occurs in a single reference. The final measurement used is the so-called “precision”, computed by dividing the number of n -gram matches by the total number of n -grams in the translation. The BLEU metric yields results that have been shown to highly correlate with that of human evaluations, through its ability to use multiple reference translations[29].

Another metric used in evaluating machine translation is NIST, an extension of BLEU[3]. The major difference between the two metrics is that the NIST evaluation adds a weight to n -grams that occur less frequently, so that they will affect the score more dramatically than the commonly-occurring n -grams. The weight for a given n -gram is calculated as follows:

$$Info(w_1...w_n) = \log_2\left(\frac{\text{the\#of occurrences of } w_1...w_{n-1}}{\text{the\#of occurrences of } w_1...w_n}\right), \quad (6.1)$$

where the weight of a given n -gram is essentially the number of times that the specific n -gram (minus the final token) occurs divided by the number of occurrences of the complete n -gram.

As opposed to these, the Word Error Rate (WER) is a relatively simple approach typically used for the evaluation of speech recognition programs[24]. It has also been applied to the evaluation of MT. It essentially calculates the Levenshtein distance between the output of a machine translation system and a single reference translation, considering each word of the candidate and reference translations as a single “character”. The distance is then divided by the total number of words in the output. The WER is rather dependent on that single translation being accurate[37].

6.6 Conclusion

The results obtained from our experiments show that the approach that we have adopted has a promising potential in improving machine translation systems.

By building a standard SMT system using our data set, we were able to obtain baseline results. By virtue of this baseline, we were able to evaluate our main hypothesis of incorporating OptPR, which has shown that at a base level it achieves comparable results, with the bonus of selecting a sentence structure that is guaranteed to occur in the Japanese language. We have also shown that by including additional post-processing steps, the overall accuracy can be improved to be greater than the baseline, and achieves more perfect matches.

Chapter 7

Conclusion and Future Work

7.1 Introduction

The field of Machine Translation (MT) is inherently plagued with various difficulties, mostly stemming from the fact that natural languages were neither designed to be understood by machines nor to be translated into every other language. In order to handle some of these difficulties, we have chosen to take a novel approach in performing MT. The goal of our research has been to evaluate the viability of using syntactic Pattern Recognition (PR) techniques in MT, and to determine the problems inherent to translation that this approach is capable of solving. In this thesis we have presented solutions to some of the issues associated with MT, outlined in the objectives below.

7.1.1 Required Data Size Reduction

Most MT systems require a large amount of data in order to perform translation reasonably well, and this data is usually specific to a given language pair. For instance, a corpus that contains English and Japanese sentences would not be applicable to English-Spanish translation. The amount of data required for accurate MT is prohibitively large for many language pairs, and such resources, simply, do not exist [39].

To reduce the size of the data set needed, we have made use of a unique representation that displays the training data as *sentence structures*, rather than the “raw” sentences. When one observes that there are less sentence *structures* than sentences themselves in a given language, it follows that less data would be needed to train a model that works with these structures.

7.1.2 Syntactic Translation Performance

Fully-fledged translation systems are typically composed of several components. While each of these components may be necessary for a given system to operate properly, a consequence of working with a natural language is that each component has an innate chance of error. We have shown that our approach has the potential to improve an important component of translation, i.e., that of reducing the amount of erroneous data fed to subsequent stages of processing.

7.1.3 Syntactic Accuracy of Translation

The best option a computer has for verifying a sentence’s grammatical correctness is to compare it to a list of every possible sentence in the language. Unfortunately, such a resource is impossible to construct. Instead, statistical methods use a language model, which is a probabilistic model that determines the likelihood of a given sentence belonging to a specific language[30]. The issue with this approach is that the language model is only an approximation and has a tendency to “over-generate”, resulting in sentences that are not grammatically correct. To correct this issue, we have replaced the language model with a list that contains a sufficient amount of sentence structures belonging to a language. While it is still infeasible to have a complete list of every potential sentence structure for a given language, we have shown that it is reasonable to have a large-enough collection to cover most of the language. Using such a resource would greatly reduce “over-generation”, and through our experimentation, we have observed no significant lack of coverage on our test cases.

7.2 Contributions

Through our research into the application of syntactic PR in MT, we have found several benefits to using our approach. The decision to use the OptPR algorithm for the purposes of translation has resulted in the development of various contributions to the field of MT, including the contribution of our own unique system. The following sections detail each individual contribution, the results obtained in regards to each contribution, and the implications that these results have on its merit.

7.2.1 Contribution 1 in Chapter 4: String Representation of Sentence Structures

As mentioned previously, the optimal way by which we can ensure that a sentence belongs to a given language is to locate it within a list of every potential sentence for that language. As a proxy for this ideal method, we have chosen to perform essentially the same operation, but instead, to compare it against a list of potential sentence *structures*. To achieve this, we have created a unique representation of a sentence structure, referred to in this thesis, as a “Template Sentence”. This representation extracts the essential meaning from a sentence, replacing every token with its Part-of-Speech equivalent.

The template sentences extracted from our training corpus were used to build both our system and a standard statistical MT system. In spite of it removing the meaning from the training data, both systems were able to translate the test sentence structures fairly accurately. Additionally, our system was able to translate the test data with many perfect and close-to-perfect matches, showing that our list of sentence structures attains coverage of the target language beyond that of our training data. Because of this finding, we believe that our system is capable of functioning without a standard language model.

7.2.2 Contribution 2 in Chapter 4: Phrase-Level Template Parsing

There are numerous challenges associated with machine translation, but one that is of particular relevance to English-Japanese translation is that of word ordering. The basic structure of English and Japanese sentences is vastly different, making it difficult to translate the structure of either language using merely elementary “re-order” operations. To mitigate this difficulty, we have used rules to break the English and Japanese sentences into roughly equivalent phrases, and to build our translation models at the sub-sentence level.

Our experiments were divided into separate phases to test the impact of implementing such an approach, and we can confirm that the accuracy of the system improved using these phrases in place of the complete sentence structures. This provided us with an additional opportunity to verify the data during the recombination stage, further improving the overall accuracy of the system as well as the number of perfect matches.

7.2.3 Contribution 1 in Chapter 5: Word Alignment Utilization

As part of the syntactic PR algorithm employed, we needed a probabilistic model similar to that of a translation model used in a standard statistical MT system. A portion of the translation model is an approximation of the likelihood that a word in the original language will become a word in the target language. We essentially need the same information, except in the reverse direction. Because of this, we have made use of existing tools for the construction of translation models, training them with our template sentence representations rather than the expected usage of “raw” sentence pairs. Additionally, we have trained the model using significantly less data than is typically used with word-alignment tools [36].

The same word alignment models were used to train both the baseline system [9] as well as our own implementation. The results obtained from running the test data against both systems demonstrate that our tool was able to discover accurate

mappings for the tokens used in our template sentences, despite the fact that the word alignment tool was not designed for our specific usage. This shows that our representation can work with existing architectures, while using significantly less data than would be needed with raw sentence pairs.

7.2.4 Contribution 2 in Chapter 5: Machine Translation using the OptPR Algorithm

The core of our research involved incorporating and assessing the viability of using syntactic PR techniques to the field of MT. The specific algorithm we have utilized is the OptPR algorithm, an algorithm that matches an input string against a set of strings [28], using probabilistic models to determine the most likely candidate. By making use of the template sentence representation and the word alignment models, we were able to construct the list and probability models required for this algorithm to function.

Our system, using the OptPR algorithm without any post-processing enhancements, achieves results comparable to that of the baseline system. The major benefit of using the OptPR algorithm is that it guarantees that the chosen sentence structure has appeared in the training data. The baseline system, on the other hand, is more capable of generating sentence structures that are close to the expected result, but are themselves not grammatically correct. With additional enhancements using the phrase-based models, our system outperformed the baseline on the test data used.

The fact that our system does not rely on a language model, but instead uses a rigid list, serves to limit the potential sentence structures generated by our system to those contained in the training data. However, as our system achieves comparable or even superior results over the baseline system, we have observed that the structures discovered in the training data sufficiently cover the structures contained in our test cases. Further, our system prevents the over-generation of grammatically-incorrect structures, while it is not dramatically affected by a potential lack of coverage over the complete set of sentence structures permissible for the target language.

7.3 Conclusion

The goal of this thesis has been to utilize syntactic PR techniques effectively in the field of MT. The capabilities of the novel approach we have taken, which uses the OptPR algorithm, has been clearly demonstrated in MT, provided that we permit a few deviations from the standard statistical MT process. The distinctions in our methods have proven to be beneficial, particularly in replacing the standard language model with our collection of sentence structures. Through this, we are able to generate accurate and valid sentence structures during MT, thus, demonstrating the viability of incorporating syntactic PR techniques in this field.

7.4 Future Work

There are many potential avenues for further research on this topic, the most pressing of which is addressing the limited coverage of sentence structures for either language. Currently, phrases are extracted using strict rules based on constituent parsing on the English side, and the particle markers on the Japanese side. Rather than imposing these rules, it would be better to train the system to discover these phrasal units automatically.

Phrase-based translation is an area of SMT that performs translation at the phrase level, and several techniques have been defined to automatically discover and align phrases in a parallel corpus. One such approach uses the same word alignment probabilities used in our research (obtained using GIZA++) [18], and so it is reasonable to believe that phrasal units could be discovered automatically using our representation. In doing so, we could then allow for a less-restrictive data set. A difference that arises between this approach and our current implementation is that, while we do extract phrases, the actual alignment still occurs at the word level. If translation was performed purely at the phrase-level using the TS representation, it could lead to an oversimplification similar to what has been observed in the third phase of our experiments. Instead, alignment could be performed at both the phrase and word levels.

While the word-based portion would function as described in this thesis, additional work would need to be performed in order to allow for higher-level alignment, specifically in regards to how the phrases are labelled. The words contained in each phrase are labelled with their PoS tags, but there is no inherent label assigned to a phrase that is automatically extracted.

This label is important, as it would then be used to discover matching or similar phrases when translating new data. A potential label would simply be determined by using the aggregate of its parts, by combining the PoS tags of the words contained in each phrase. Alternatively, work has been done to cluster phrases in order to discover groupings of interest[20]. This research differs from what we have done because the clustering is performed on “raw” words rather than on the consequent tags. However, the same basic idea of grouping similar phrases into categories could allow us to work with phrase-types, similar to what has been done in this thesis. Alignment could then be performed at the phrase-level, allowing us to mitigate major word ordering differences. In extracting and handling phrases through automated methods, the grammar that the system is allowed to handle can be greatly enhanced, with the major concern now being how *well* it could handle different sentence structures.

A potential issue with being less restrictive with the training data is the higher likelihood of there being errors in the sentence alignments contained in the parallel corpus. However, research regarding the impact of such errors has been performed, and it has been found that phrase-based translation approaches are highly resilient to this phenomenon[13].

Finally, further research could be performed in defining a more suitable tagset for our problem. The representation we have used for sentence structures essentially replaces all instances of words with their respective syntactic-class, using PoS tags. In this regard, we have used the Penn-Treebank tagset due to the fact that taggers are readily available for this tagset. Developing a custom tagset that would better facilitate our method of translation could potentially improve its accuracy.

Bibliography

- [1] Jason Baldridge. The OpenNLP Project, 2005.
- [2] C. Brockett, T. Aikawa, A. Aue, A. Menezes, C. Quirk, and H. Suzuki. English-Japanese example-based machine translation using abstract linguistic representations. In *COLING-MTIA '02 Proceedings of the 2002 COLING workshop on Machine translation in Asia*, volume 16, pages 1–7, 2002.
- [3] George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *2nd International Conference on Human Language Technology Research*, pages 138–145, 2002.
- [4] B. J. Dorr, E. H. Hovy, and L. S. Levin. Machine translation: Interlingual methods. In *Encyclopedia of Language and Linguistics 2nd edition*, 2004.
- [5] Bonnie Dorr. Parameterization of the interlingua in machine translation. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*, 1992.
- [6] T. Mitamura E. Nyberg and J. Carbonell. The KANT Machine Translation System: From RD to initial deployment. In *Proceedings of LISA Workshop on Integrating Advanced Translation Technology*, pages 3–4, 1997.
- [7] T. Erjavec, K. H. Sangawa, and I. S. Erjavec. jaSlo, a Japanese-Slovene learners’ dictionary: methods for dictionary enhancement. In *XII Congresso Internazionale di Lessicografia*, pages 6–9, 2006.

- [8] T. Erjavec, K. H. Sangawa, and I. S. Erjavec. jaSlo: Japonsko-slovenski viri za učenje japonscine, September 2013.
- [9] P. Koehn et al. Moses: open source toolkit for statistical machine translation. In *ACL '07 Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180, 2007.
- [10] Trang Ho et. al. Tatoeba: Collecting Example Sentences, 2014.
- [11] M. Federico, N. Bertoldi, and M. Cettolo. Istm: an open source toolkit for handling large scale language models. In *Interspeech*, pages 1618–1621, 2008.
- [12] T. Fujinami and C. Nanz. The 101 Translation Problems between Japanese and German/English. Technical report, University Stuttgart, 1997.
- [13] C. Goutte, M. Carpuat, and G. Foster. The impact of sentence alignment errors on phrase-based machine translation performance, 2012.
- [14] D. Hindle and M. Rooth. Structural ambiguity and lexical relations, 1993.
- [15] John Hutchins. The first public demonstration of machine translation: the Georgetown-IBM system, 7th january 1954, 2005.
- [16] John Hutchins. Towards a definition of example-based machine translation. In *Proceedings of Example-Based Machine Translation*, pages 63–70, 2005.
- [17] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.
- [18] P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54, 2003.
- [19] Phillip Koehn. *Statistical Machine Translation*. Cambridge University Press, 2010.

- [20] D. Lin and X. Wu. Phrase clustering for discriminative learning. In *Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, pages 1030–1038, 2009.
- [21] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.
- [22] M. P. Marcus, M. A. Marcinkiewicz B., and Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.
- [23] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.
- [24] A. C. Morris, V. Maier, and P. Green. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition. In *INTERSPEECH*, 2004.
- [25] Makoto Nagao. A framework of a mechanical translation between Japanese and English by analogy principle. In *Proceedings of the international NATO symposium on Artificial and human intelligence*.
- [26] R. Navigli and P. Velardi. Structural Semantic Interconnection: a knowledge-based approach to Word Sense Disambiguation. In *Pattern Analysis and Machine Intelligence, IEEE Transactions*, volume 27, pages 1075–1086, 2005.
- [27] F. J. Och and H. Ney. Improved statistical alignment models. In *ACL00*, pages 440–447, Hong kong, China, 2000.
- [28] B.J. Oommen and R. L. Kashyap. A Formal Theory for Optimal and Information Theoretic Syntactic Pattern Recognition. In *Pattern Recognition*, volume 31, pages 1159–1177, 1998.
- [29] K. Papineni, S. Roukus, T. Ward, and W. Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, 2002.

- [30] A. Ramanathan, P. Bhattacharyya, and M. Sasikumar. Statistical Machine Translation. Technical report, Indian Institute of Technology, Bombay, 2008.
- [31] W. B. Dolan S. D. Richardson and L. Vanderwende. MindNet: acquiring and structuring semantic information from text. In *COLING '98 Proceedings of the 17th international conference on Computational linguistics*, volume 2, pages 1098–1102, 1998.
- [32] S. Sato and M. Nagao. Toward memory-based translation. In *COLING '90 Proceedings of the 13th conference on Computational linguistics*, volume 3, pages 247–252, 1990.
- [33] D. Searls and S. Dong. A Syntactic Pattern Recognition System For DNA Sequences. In *2nd Intl. Conf. on Bioinformatics, Supercomputing, and Complex Genome Analysis*, volume 2, 1993.
- [34] Yasuhito Tanaka. Compilation of a multilingual parallel corpus. In *PACLING 2001*, pages 265–268, 2001.
- [35] Gregor Thurmair. Comparing rule-based and statistical MT output. Proceedings of the amazing utility of parallel and comparable corpora, 2004.
- [36] L. Tian, F. Wong, and S. Chao. Word Alignment Using GIZA++ on Windows. In *Machine Translation Summit*, volume 13, pages 369–372, 2011.
- [37] Hans Uskoreit. Survey of Machine Translation Evaluation. Technical report, Saarland University, 2007.
- [38] S. Vogel and A. Waibel. Improving Statistical Machine Translation in the Medical Domain using the Unified Medical Language system. In *Coling 2004*, 2004.
- [39] H. Wu and H. Wang. Pivot language approach for phrase-based statistical machine translation. In *Machine Translation*, volume 21, pages 165–181, 2007.
- [40] H. Xiong, W. Xu, H. Mi, Y. Liu, and Q. Liu. Sub-sentence division for tree-based machine translation. In *ACL-IJCNLP 2009 Conference Short Papers*, pages 137–140, 2009.

- [41] K. Yamada and K. Knight. A syntax-based statistical translation model. In *ACL '01 Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530, 2001.