CONTEXT-DRIVEN ENGLISH TO JAPANESE TRANSLATION

・Johnathan Dewey・Furman University・B.S. Computer Science・

(1) Abstract

In past decades, computational translation has been a widely used tool due to its accessibility and ease of use. Despite its popularity, computational translators are second-rate compared to multilingual humans who can provide natural translations of phrases—which computational translators cannot. This paper presents algorithms and techniques of a mechanical translator that uses context to drive the transfer of phrase representation from English to Japanese thus effectively producing a translation. This translator consists of three distinct steps. The first takes raw text in English, it is tokenized, and each token is then classified according to its part of speech. These tokens act as terminals in a sentence tree that is constructed using a bottom-up technique. Given a sentence tree, the next step works to manipulate the internal nodes (non-leaf) nodes of the sentence tree to reflect an ordering of nodes consistent with the sentence structure Japanese language: Subject-verb-object in English to subject-object-verb in Japanese. The last step involves manipulating the terminal nodes in the sentence tree so that the translator 'translates' English words to Japanese, verify conjugation, and that the use of particles is correct. We verified the effectiveness of the proof-of-concept translator by testing for usage of proper nouns, conjunctions, and all variations of tense features.

(2) Key Words

English and Japanese Translation, X-Bar Theory, SLIM Theory, Linguistic Corpora

(3) Introduction

Since computational translation is an inaccurate and inefficient tool, it is important to theorize a translator that uses context of language to aid its decisions in translation. words of a language are normally broken into distinct parts of speech (POS) which are classes of lexical items that have a specific function within the language. Sentence formulation is broken into three distinct parts which determine the sequence in which certain words are uttered: subject, objects, and verbs. These two observations of speech are good indicators of context that can be used by a translator to make decisions during the translation process.

The context-driven translator introduced in Figure 1 works in three divided phases with different related sub-algorithms. The workflow involves the building and modification of a syntactic sentence tree to derive Japanese translations. We present a brief walkthrough of the workflow provided in Figure 1 and its components.

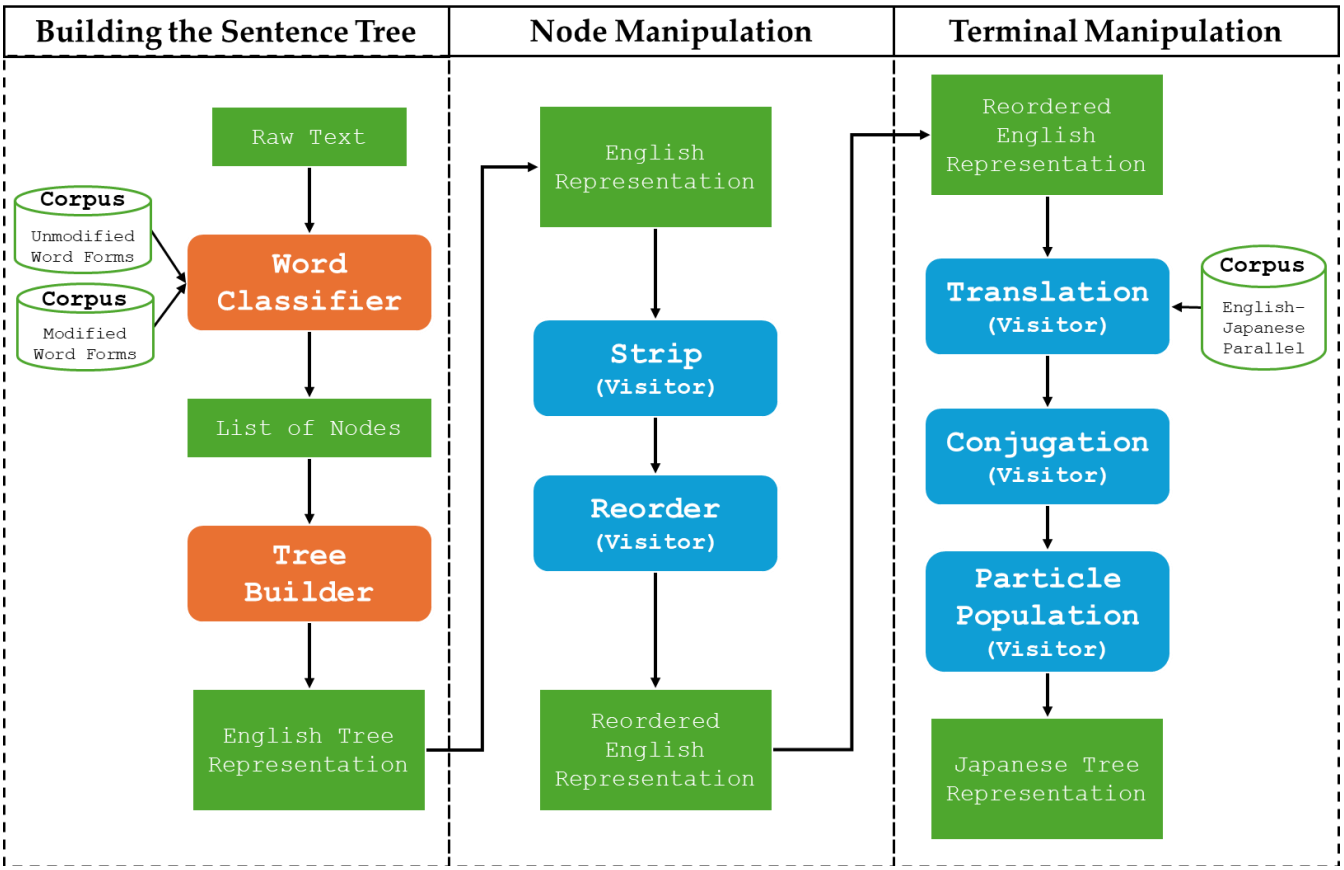

Figure 1: The workflow for context-based English to Japanese translation using sentence trees.

Step 1: Sentence tree construction. The goal in this first step is to construct a tree data structure representation of an English sentence through the recognition and encoding of individual words. It starts with the classification module which takes text input and encodes it into nodes. The classifier accomplishes this using a corpus database [1] that contains English words with their respective POS. The resulting nodes represent the different POS in the sentence.

Then the nodes are passed to the sentence tree constructor in the form of a root node for bottom-up construction. The root node, which is the tense phrase node, initializes the recursive separation and encoding of nodes into the tree. The resulting sentence tree has nodes that represent the hierarchy of phrases, and terminals that represent the words of a sentence.

Step 2: Node Manipulations. The goal in the second step is to manipulate the nodes of the sentence tree to match the syntactic representation of a Japanese phrase. All the following functionalities are implemented using a visitor design pattern [2] to modify the sentence tree. This node manipulation step begins with a visitor we call a "stripper" (or strip visitor) that detects unsupported words in Japanese and removes them from the tree; subtrees of those nodes are then reconnected to ensure a well-formed sentence tree. Then another visitor, a "reorder" visitor, passes over the sentence tree to detect any subphrases that

are out of place in the context of Japanese sentence order, and places them in the correct location in the tree.

Step 3: Terminal Manipulations. The goal of the final step is to manipulate the words encoded into the terminals of the tree, so that the text can match Japanese words. Once again, the functions use a visitor pattern to modify the tree. It starts with the "translation" visitor where a terminal's text is sent to a parallel corpus database and gets matching Japanese words. Next, is the "conjugation" visitor, where a terminal's conjugation of POS is verified. Finally, the "particle" visitor visits each terminal and makes sure that the correct Japanese particles, which function as linguistic markers, are present.

(4) The Theoretical Frameworks of The Translator

Computational Translation, or Mechanical Translation is an approach to the task of translation that uses specific computational tools to achieve a transfer of text or speech from one language to another. Like other computational disciplines, it tries to mimic the previous work of humans in a particular field of study. However, compared to human translation, computers perform relatively worse and miss out on the translation of linguistic discrepancies between language—which is intuitively handled by multilingual humans [3]. So, the problem is how can we get to the point where a translation system can achieve these translations? This is a notion that many scholars disagree on.

Computational Translation now exists in an interesting state of contention where scholars are perpetually divided between the process of Statistical Machine Translation (SMT) and Neural Machine Learning (NMT)—with many defending the use of NMT over SMT [3] [4]. The critiques of both are both in its use of data. SMT uses corpora, or knowledgebases of lexical information, to facilitate translation, which its efficiency is critiqued as being limited to the quality of the corpora. On the other hand, NMT uses Machine Learning (ML) to train models on lexical data to create relationships between words of different languages, which is critiqued for being limited to the quality of the test data. Also, NMT is very "data-hungry," and mines internet-accessible data to train its models [4]. This "data-hungry" model is not remarkably effective for languages without media applications [4], whereas SMT can if the corpus is well-formed.

However, both methods are valid as they can be used in conjunction to solve different sub-problems of translation. Therefore, the scope of this research relates to the implementation of an SMT context-driven translator to provide accurate translations. The translator will focus on the translation of text from English to Japanese. Both languages are relatively diverse with different linguistic attributes, and it is theorized that SMT can use context to effectively solve discrepancies between the languages.

Since translation is a task that belonged solely to linguistics before the discipline of computational translation, it would be inappropriate not to incorporate linguistic frameworks into a translator. Thus, a translator would be refined in its actions instead of using product-driven mechanisms to achieve translations. For this, the translator uses the linguistic syntax theory of X-Bar as the framework [5] for

structuring and organizing lexical items in a phrase. Like SMT, X-Bar is very contentious among linguistics, and this project intends to prove that X-Bar is a useful concept of phrase representation that is effective in the translation of language over other syntax theory.

The following sections describe the tools and concepts necessary to create an efficient and accurate translator.

(a) SLIM Framework

Previously, we introduced the concept of computational translation and its lacking to match the effectiveness of human translators. This pitfall in mechanical translation's ability to produce natural translations implies that translation is one of the few tasks in which humans outperform computers. This fact asserting outperformance is explained by Hausser, [6] in which this pitfall is not due to the state of computing, but to the fact that—unlike computation—linguistics is yet to find a "comprehensive, verifiable, functional theory of language."

SLIM Theory		
Acronym	**Principle**	**Attributes**
(S) - Surface Compositional	Methodological	Syntax composed of concrete word forms
(L) - Linear	Empirical	Grammar recognizing sequential utterance of phrases
(I) - Internal	Ontological	Treatment of language as a cognitive process existing in speaker and hearer's minds
(M) - Matching	Functional	Contextual decisions that switch between literal and contextual meanings

Figure 2: The components of SLIM Theory, as described by Roland Hausser to be a "functioning, mathematically precise, and efficient theory" for mechanical translation [6].

Essentially, computation disciplines are uniform in theory which has been verified through scientific and mathematical proof, while linguistics does not have a unifying cohesive theory that explains the behavior of all languages [6]. However, explaining the behavior of all languages is an impossible task as language is one of the most variable concepts to exist. Each language exhibits themselves differently with diverse linguistic attributes. This incohesive nature of linguistic theory is why many theories are constantly critiqued or dismissed—as we will see with X-bar Theory.

The intent of a computational translator should not be one that treats language as a uniform task but tries to encode the unique representations of different languages and translates through algorithms catered to the specific translation of two languages. In this translator we cater on the translation of English to Japanese in recognition of this notion.

Despite the variability of language, scholars have theorized techniques for making translation more efficient. Provided in Figure 2 is the *SLIM Theory*, which Hausser [6] claims is a "functioning, mathematically precise, and efficient theory" of mechanical translation [6]. Of the four principles in Figure 2, it is valuable for Hausser to include the ontological principle which treats translation as a cognitive process between humans—and not simply an abstract task capable by anyone or anything. Through its inclusion, it acknowledges the innate human ability of speech formation, and that a computational translation system would mimic these human behaviors. Additionally, the inclusion of a functional

principle, which recognizes the nature of a word having literal and contextual meanings, forces the development of a translator sensitive and driven by context.

SLIM Theory is implemented in this translator to act as a framework that will produce the natural translations desired. It is important that mechanical translation software uses linguistic concepts since it is essentially its child in the scope of history. Centuries of research has gone into linguistics, and it would be foolish to ignore it. Therefore, it is impossible to make accurate algorithms that operate on language without studying the behaviors of such languages.

(b) SMT vs. NMT

There exists a noticeable schism in the computational translation's schools of thought. On one hand, there are advocates for Statistical Machine Translation (SMT) which employs a linguistic corpus which is a large database that contains linguistic lemmas and data such as its frequency in specific texts. Therefore, SMT uses empirical data on the frequency and context of words to provide a more cohesive and context-sensitive translation [7]. On the other hand, there are advocates for Neural Machine Translation (NMT) which uses Machine Learning (ML) techniques to train a model to learn and predict the equivalent meaning of phrases between languages. With fervorous advancements among ML, many think that it is objectively better. This is not necessarily the case, as SMT has many attributes that make it an effective method of translation.

SMT's attributes are especially useful in the scope of solving context-dependent translation problems. The main feature of SMT is a linguistic corpus that organizes words of a language and relevant linguistic information. One instance of relevant linguistic information is the POS tagging that enables the organization of lexical items into their respective parts—such as nouns, verbs, adjectives, etc. POS tagging provides richer contextual information for a sentence and improves decision making in translation [8]. Additionally, POS tagging enables systems to interpret the structure of sentences without knowing the exact translation [8]. Essentially, regularities in phrase structure and POS can be used as context and can make decisions to attain translation without knowing the individual words. This fact is emphasized in the translator where the meaning of words is not needed to drive translation. In fact, translation of words after a Japanese phrase structure is derived is the only instance where a word meaning is needed.

However, there are some detriments to POS tags, as certain POS can often be redundant and can hinder decision making. POS, like nouns, are known to be strong in aiding decision making, while redundant words such as determiners impede the decision making of translation [8]. However, it is theorized in this translator that by treating certain POS as more important through a precedence hierarchy, then the instances of hinderance in POS tagging can be minimized. In such a hierarchy POS like verbs and conjunctions are treated as superior in contextual meaning than that of a noun or determiner. Therefore, representations of phrases can easily be discernable from the existing POS in the phrase.

When comparing SMT and NMT it is hard to determine if one is objectively better than the other because they both yield accurate results. However, both techniques are known to have their own detriments such as ill-defined test data of ML models in NMT and the management and expansion of a corpus in SMT. However, when it comes to translating a simpler abstraction of language, such as a programming language, it becomes clear that one technique outperforms the other. According to Phan and Jannesari [9] whom employed the use of 2 SMT's and 2 NMT's to translate source code from one programming language to another, they discovered that translation yielded a percent accuracy of 60-90% from SMT and 59-83% accuracy from NMT [9]. From these results they asserted that when translating source code that SMT outperforms NMT in their current state [9]. This fact proves that SMT is a relevant approach to mechanical translation, and if SMT can outperform NMT for a simpler abstraction of language, then it is theorized that a natural language translator will yield accurate translations if it employs a SMT system.

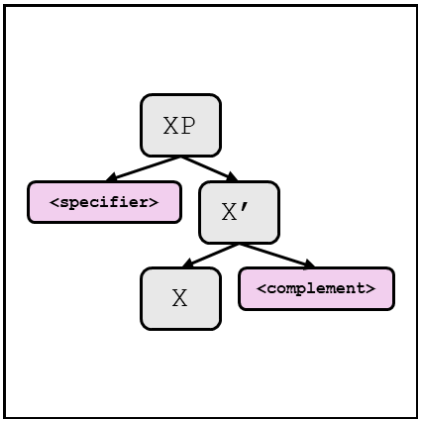

Figure 3: Structure of the X-Bar phrase node [10]. The variable X is the head or POS (Noun, Verb, etc.) that carries the most meaning. the X-bar (X') establishes a close connection between X and its complement. The specifier are phrases that come before the head.

(c) Syntax Trees and X-Bar Theory

Mechanical translation requires computational operations to act as processes in the task of translator. These operations should be efficient and can be efficient if the representation of phrases to be translated are organized in an efficient and logical manner. The linguistic field has produced many frameworks pertaining to the representation of phrases as syntactic trees that are built from phrase rules in a generative grammar.

Syntax trees in linguistics are the hierarchical organization of lexical items within a phrase, and a traversal of the sentence tree produces the sequential utterance of lexical items of a phrase. However, there have been many competing theories of generative grammar—and linguists disagree on most of them. The critique of generative grammars derives from their perceived complexities and limitations [5]. Its complexity is not referring to its computational complexity which trees perform particularly well. Rather, complexity refers to the complex grammar rules that are needed to provide well organized sentence tree structures. If we treat translation as a solely human task, this notion against generative grammar is agreeable. However, the conditional-based rules of generative grammar are a task computer exceed in. As for the limitations of syntax trees, they can only do so much in the usage of irregular language. Syntax trees can organize the lexical items of a sentence well, but in the context of translation, it can only produce natural translations for grammatically correct phrases. This limitation is true; however, these cases of irregular language are perfect instances to integrate NMT to help supplement the areas that SMT lacks.

Despite the critiques of generative grammar, the representation of sentences as a hierarchical tree is valid in the context of translation. The translator uses X-Bar Theory which is a theory of syntax that creates representations of nuanced phrases (See Figure 3). X-bar aims to make stronger predictions as to what constitutes a grammatical sentence [10]. Their key feature, the X-bar node, is an extra node of abstraction that permits nuanced or "non-flat" phrase structures so that phrases can be independent of each other yet have associations [10].

Another feature of X-bar theory is the assumption that phrase structure is identical for all POS and all languages. While a bold assumption, it is based on some truth. Many languages share a commonality in phrase structure where the head (lexical item carrying meaning) of a sentence tree shares a more intimate relationship with their complement than specifiers [10]. This fact is reflected in the structure of an X-phrase (XP) node where the first level of the phrase includes a specifier ("I" in "I went home"), and the second level includes a complement ("home" in "I went home"). The complement is placed next to the head ("went" in "I went home") of the phrase to preserve this close relationship, and therefore has the specifier come before the head.

In the context of computation, X-bar can be expressed as binary tree structure [5], which is beneficial to this translator because it will produce efficient algorithms of translation that can perform on and store information efficiently.

(5) Satisfying Translation Frameworks and Research Objectives

The objective of this translator is to experiment with SMT techniques in conjunction with an X-bar phrase representation to produce accurate English to Japanese translations. The translator uses the SLIM theory as a framework to bring this translator to the standard of a functional and mathematically precise mechanical translator. Figure 4 briefly explains how each principle of SLIM theory is satisfied through the translator's components to summarize the assertions made in this section.

Satisfying SLIM Theory	
Principle	**Reflecting Component**
Methodological	• Corpus datasets to store lexical information • Syntax tree as the main representation
Empirical	• X-Bar Theory's organization of nodes using a head
Ontological	• X-Bar Theory's universality assumption of phrase organization
Functional	• Lookups into corpora to get literal meanings • Tree traversals to get neighboring lexical items for the contextual meaning

Figure 4: Reflection of the SLIM theory in the context of the tools used for the translator. "Reflecting Component" details how each principle from Figure 2 is satisfied through the translator's components.

The first principle [6] of SLIM that needs to be satisfied is the methodological principle of a syntax composed of concrete word forms. The use of SMT and corpora will aid translation by having a cohesive and efficient store of lexical knowledge available. In this translator, 3 main corpora are used by components within the translator, where one of which stores concrete word and modified word forms of English [1], as well as equivalent lexical items between English and Japanese.

Furthermore, the use of syntax trees satisfies the need for a cohesive syntax, and the X-bar structure supplements this need through a structure that organizes nuanced phrases in a universal manner.

Second, is the empirical principle of a sequential progression of words within a sentence [6]. Once again, X-bar can satisfy this principle. The concept of the head, which is the "X" in Figure 3, allows not only for the sequential ordering of words, but the sequential ordering of constituent phrases. The generative grammar of X-bar understands and incorporates tendencies of nuanced speech. Like in the X′ node in Figure 3, the "non-flat" phrase representation establishes that a head and its complement share a close relationship, whereas the head and the specifier in the XP node in Figure 3 share a weaker relationship. Therefore, X-bar grammar is one that logically orders phrases efficiently, and a traversal of the resulting sentence tree would reveal the correct sequence of words.

Third, is the ontological principle of recognizing language as a cognitive process that exists between speakers [6]. The assertion made by X-bar theory claims that their phrase representation is the same in all languages [10]. They also claim that it mimics human behavior due to the introduction of the X' node that creates relationships between the head, complement, and specifier which are meant to represent how humans construct phrases [10]. Therefore, X-bar is assumed to satisfy the ontological principle due its satisfaction of recognizing sentential representation from a human cognitive aspect, and the cognitive assumption that derives from the assertion of being a universal representation for phrases in all languages.

Finally, the last principle is the functional principle [6] of recognizing the concrete and contextual meanings of words. This is satisfied through SMT and X-bar. Once again, the main assistance from X-bar manifests in its specific phrase structure. Upon searching for the contextual meaning of a head in a XP, there is an expected and short path to the contextual modifiers of a head. This makes it so that switching between contextual and concrete word meanings are simple and inexpensive operations. SMT assists the translator through POS tagging and word searches into the corpus datasets. The lookups for specific words into a parallel corpus, which contains one-to-one equivalent words between English and Japanese, can achieve a word's translation in efficient time. Also, POS tagging allows for each node to belong to a specific class of nodes. In this translator, nodes are divided into phrase nodes, bar nodes, and terminals which have unique implementations for all POS in a language. Using type referencing for conditional statements is an efficient way to figure out what a node is without altering the structure of the whole implementation. Therefore, individual translation algorithms can access these nodes and make context-dependent decisions.

(6) The Translation Algorithm

Now that we have established the general workflow of the system, we provide a walkthrough of the system in Figure 5 and its individual components. The sentence used in this example is "She found a

job last Friday" which translates into Japanese as *watashiwa saigono kinyoubini shigotowo mitsuketa* (私は最後の金曜日に仕事を見つけた).

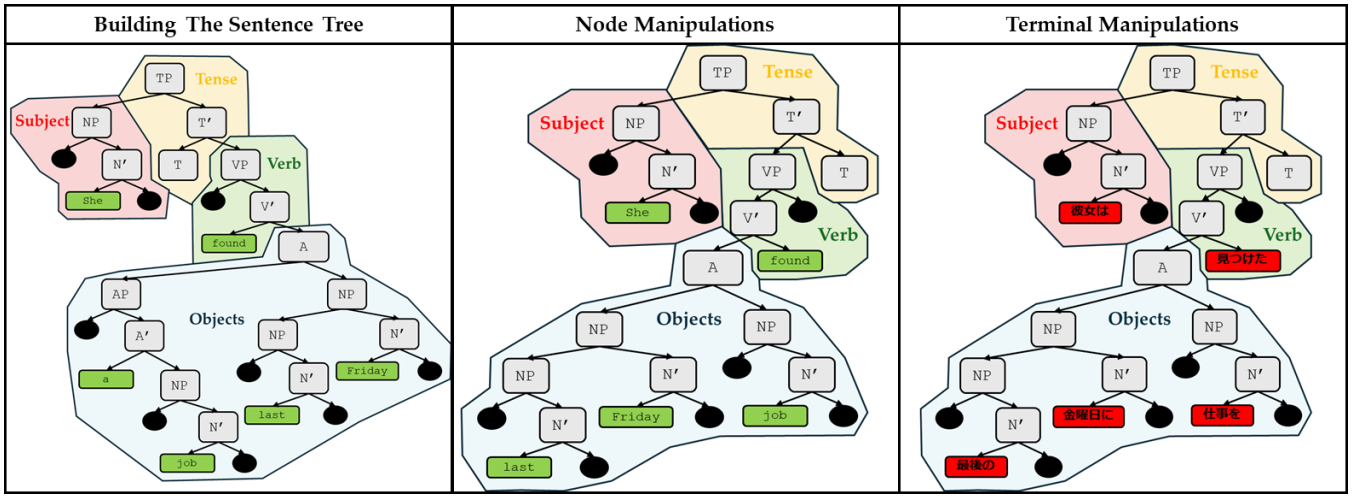

Figure 5: Overview of the context-dependent translation system. Process is split into three phases in which each phase works toward achieving a phrase representation of the target language Japanese. Used for reference throughout the introduction.

(d) Preprocessing and Encoding Text into Nodes

The first stage of the translator are the components that set-up and verify the nodes to be used in the sentence tree. In the first part of preprocessing, text discrepancies are recognized and doctored so that it will not cause errors for a search into the corpus. Such discrepancies include the inclusion of contractions to exhibit tense features. For example, words like "won't" are a contraction of the tense feature "will" and a negative "not." Such contractions do not exist in Japanese, so instances of contractions are split up into two distinct lexical items.

Following preprocessing, each word of a phrase is sent to the primary corpus for a lookup. The primary corpus [1] contains the lexical information of English words and corresponding lexical information like POS (see Figure 6). However, the primary corpus data contains only the lexical information for concrete word forms. For modified word forms like "went" (modified from "go") or "working" (modified from "work") they will fail the initial lookup and perform another search into the

Primary Corpus Data	
English Word	**POS**
finish	Verb
slow	Adjective
early	Adverb

Secondary Corpus Data		
Modified Form	**Concrete Form**	**POS**
finished	finish	Verb
slowest	slow	Adjective
earlier	early	Adverb
mornings	morning	Noun

Figure 7: Secondary English Corpus that contains modified word forms and maps them directly to their unmodified form with their corresponding POS.

secondary corpus [1] (see Figure 7). This corpus contains modified word forms and maps them directly to their unmodified word forms. Therefore, if an English word exists in the corpus, they it can be represented fully in the resulting tree. However, lexical items like proper nouns are not in the corpus, so the translator assumes that unfamiliar words are nouns.

However, testing iterations of the translator revealed discrepancies in word encodings. For continuous forms of verbs such as "working" it would be encoded as an adjective. To solve such an issue, the translator implements a context verifier that ensures that POS makes sense contextually. For example, Adjectives must reference a noun or reference another adjective. In cases where such requirements are not met, the nodes are re-encoded to find the correct POS that should exist.

(e) Building the Sentence Tree

The purpose of the preprocessor was to provide the terminals representing words of a phrase, which are then used as the building blocks of the sentence tree. In the next phase of the algorithm, the creation of a sentence tree using context derived from the type of nodes present is accomplished. The sentence tree defined for this translation algorithm is one that uses the X-Bar node representation that allows for phrases to have a specifier and complement if necessary. The resulting tree, as shown in Figure 9, stores the subject, verb, and objects as children of the tense features in a sentence. The structure uses vertex nodes (`NP`, `N'`) to create hierarchical relationships between phrases, and uses terminals to represent the lexical items of a sentence.

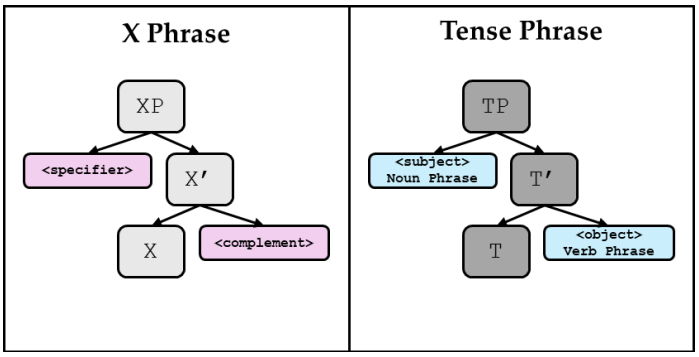

Figure 8: Comparison between the X Phrase (XP) structure and the Tense Phrase (TP) structure. See Figure 3 for XP information. TP captures the tense features and modal auxiliaries ("can," "should," etc.). These features are encoded in the Tense (T) terminal. Like the X' node, T' establishes a close connection between tense and the verb phrase, whereas TP exhibits the placement of a subject before a Verb phrase.

(1) Building the Tense Phrase

In this first step we make a sentence tree out of a list of nodes. These nodes are sent to the root node of a sentence tree which is the `TP` node (see Figure 8). In X-Bar Theory, `TP` is a phrase node that incorporates the requirements for a complete sentence.

A `TP` node has a left child that represents the subject of the sentence. The right child is known as the T-bar (`T'`) node and allows for its right child to contain a verb phrase that represents the verb and object of a sentence. The left child, however, is known as the tense (`T`) node because it stores all the tense features, such as present/past or modal auxiliaries, of the sentence. Theoretically, the `TP` node is an overloaded `XP` node—evident from Figure 8. This is because the positions where the subject and verb-object phrases exist in a `TP` are the locations where specifiers and complements exist in a regular `XP`.

In the TP, the list is separated and passes the nodes containing the subject (“She”) of the sentence to the left child of TP—where the subject subtree is encoded. In this case, the subject is a simple noun phrase containing the pronoun “She.” The rest of the nodes (“found,” “a,” “job,” “last,” “Friday”) go to the right child which is the T′. In T′ the tense information of the sentence is recorded in its left child which is the T terminal.

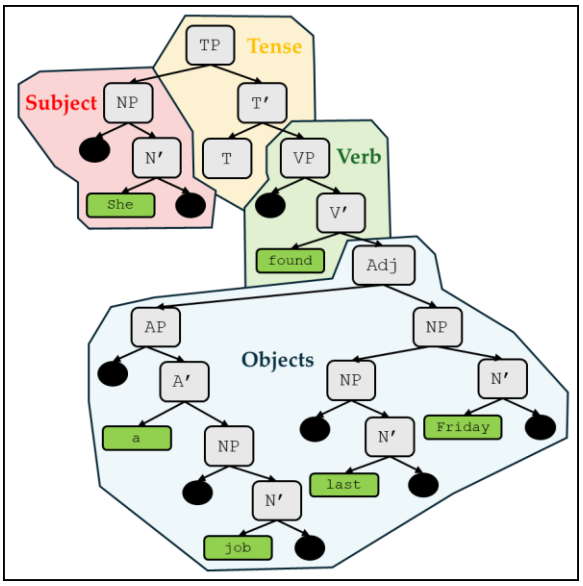

Figure 9: The building of the English sentence tree for the sentence “She found a job last Friday.”

After the recording of tense information, the same nodes (“found,” “a,” “job,” “last,” “Friday”) are always sent to verb phrase (VP) so that the verb (“found”) can be separated from the objects (“a,” “job,” “last,” “Friday) of a sentence. We know it is a VP because the X-Bar structure of nodes states that the right child of T′ is the VP required for a complete sentence [10]. Therefore, we are in VP and need to discern the location of the verb in relation to the remainder of the sentence. the verb “found” does not have any specifiers, or words preceding it, so we can pass on to the verb bar (V′) level. V′ separates the verb terminal and the remaining sequence of nodes and encodes “found” as the V’s left child. Once the verb is encoded at the V′ level, the remaining objects are sent to an adjunct (Adj) node.

(2) Building the Adjuncts

The adjunct node acts as a separator between constituent phrases so they can remain separate subphrases. The direct object of the verb “found” is “a job,” so this constituent phrase is the left child of the adjunct so that it is next to the verb in the context of a sentence tree’s traversal. This position is known as the direct object because it is the complement of the verb in the sentence. The phrase “last Friday” is an adjunct phrase, so it goes to the right child of the adjunct node so that it comes last in the context of a sentence tree’s traversal.

These constituent phrases that reference the adjunct node are determined through an adjunct identifier, which breaks down individual constituent phrases. For the case “a job” it consists of an article and a noun which is recognizable constituent phrase to the identifier. Similarly, the case “last Friday”

POS Precedence Order		
Rank	**POS**	**Word Example**
1	Conjunction	"and," "but," "or"
2	Verb	"went," "have," "worked"
3	Preposition	"at," "on," "from"
4	Adverb	"lately," "gently," "across"
5	Article	"the," "a," "his," "my"
6	Determiner	"either," "neither," "fewer"
7	Negative	"not," "-n't" (closed class)
8	Noun	"park," "car," "Steve"

Figure 10: Precedence of nodes in terms of identifying a head for an XP phrase. POS are ordered based on magnitude of meaning. Conjunction is the most important because it connects large phrases which would benefit from a higher position in the tree hierarchy. Conversely, Nouns are on the bottom because any phrase can have innumerable nouns—which may result in noisy trees without this precedence structure.

recognizes the sequence of two nouns as a constituent phrase. These phrases are encoded as phrases separately, so as to not put them directly under the same hierarchy and inadvertently make the phrases dependencies of each other.

The individual constituent phrases are encoded using the precedence of certain POS to determine the representation of the constituent phrases. These precedencies defined for POS in the translator are reflected in Figure 10. Realistically, a sentence tree with X-bar is susceptible to right leaning trees if each node is treated with the same importance. With a precedence feature, sequences can be broken down into well-organized nodes. For example, "a tough job" would be recognized as an adjective phrase with a noun phrase (`NP`) complement rather than a `NP` with an adjective specifier. The reason for organizing phrases as such is that most phrases are innately `NP` because most phrases describe nouns. However, through this algorithm's precedence search, it can make more context rich decisions rather than the default POS context always being a noun.

Therefore, using this precedence search, the constituent phrase "a job" gets encoded as an article phrase (`AP`) ("a") with a `NP` ("job") complement. Similarly, the constituent phrase "last Friday" gets encoded as a `NP` ("Friday") with a `NP` specifier ("last"). These constituent phrases are then made into the left and right child of the adjunct node, which effectively separates them.

(f) Processing the Sentence Tree

The purpose of building a sentence tree was to provide a basis for organizing a sentence's structure in a logical and uniform matter. The remainder of the translation algorithm then manipulates the sentence tree to conform to the structure of a Japanese sentence. These manipulations are broken into specific algorithms that all implement a unique visitor pattern. Visitors are a type of design pattern for object-oriented systems—such as a tree populated with nodes--which emphasize flexibility [11]. They allow for the definition of many unrelated functions to be defined without ever changing the classes of an object [11]. The ability to define functions without modifications supports the open-closed principle for clean code, which allows for extensions of a system without modifying the current implementation. In this algorithm visitors are used to transverse the sentence tree and make decisions to the structure of the sentence tree and the information held in its nodes. The main advantage aside from the open-closed principle is that visitors, through accept methods in an object's class, remove the need for frequent

recompilations of an object's class and frequent typecasts to determine classes [11]. Therefore, utilization of visitors is sure to improve the overall complexity of the algorithm.

(3) Node Manipulations: Node Strip and Reordering Visitors

After the sentence tree is constructed, the translator manipulates multiple nodes to (1) remove unnecessary lexical items and (2) reorder the sentence tree to achieve a Japanese word order—relying on POS context to make decisions. Through these manipulations of the tree, we can eliminate discrepancies between English and Japanese sentence structures before translating the tree. Essentially, it is a form of preprocessing for the upcoming word translation algorithms that come after these node manipulations. The resulting sentence tree is one that represents an English sentence in Japanese SOV sentence order.

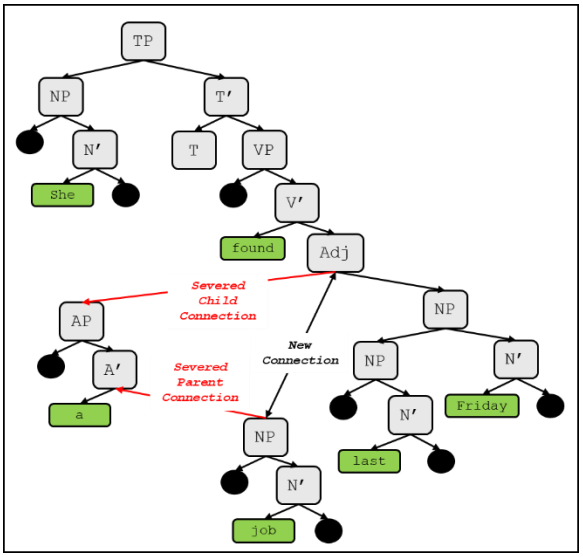

Figure 11: Node Strip Visitor: Deletes article phrases and verb phrases that do not have equivalents in Japanese linguistic properties. The article phrase (AP) is detected by the visitor and connects the nested noun phrase (NP) to the parent of AP while severing previous connections of AP.

The first visitor which searches and eliminates unnecessary lemmas from the SL for the TL is the node strip visitor. This visitor exists due to a common discrepancy between phrase representation in English and Japanese. Essentially, Japanese lacks a clear future tense, while in English words the word "will" is very indicative of an action happening in the future. In Japanese, future tense is not discernable from verb conjugations and use the same conjugations as present tense. Therefore, in terms of conforming to the Japanese sentence structure, we remove any words that do not exist in the same context as English words. Similarly, for articles, there is no definite ("the") or indefinite ("a") article for nouns in Japanese. Therefore, these unnecessary words must be removed from noun phrases. Negligence to remove inequivalent words from the structure of the sentence tree will result in a noisy sentence tree with discrepancies. Therefore, the strip visitor will result in a sentence tree that reads "She found job last Friday."

However, it is pertinent to discuss the removal of words that exist in both English and Japanese but result in a noisy sentence tree. In English, the verb "to do" can exist independently in phrases like "do your homework." However, the verb can also exist in conjunction with other verbs—especially with negative modalities—in phrases like "I do not want anything." In Japanese, "I do not want anything" would not include the lemma "do" as a separate entity because the verb "wants" expresses "do" implicitly and "not" is expressed as a conjugation of the verb. Therefore, some sentences initially produced noisy Japanese trees with confusing combinations of a verb with "do" and "not."

After the complete removal of noisy lemmas, the sentence tree is then passed on to the reordering visitor. Sentence word order is arguably the largest discrepancy between English and Japanese translation. Objectively, English has a "Subject-Verb-Object" (SVO) ordering of phrases, whereas Japanese has a "Subject-Object-Verb" (SOV) ordering of phrases. This means that the structure of nodes and their children must match the traversal of a Japanese sentence tree. To accomplish this, the visitor visits each node and reorders the node's children based on encoded instructions. For example, upon visiting any verb, the visitor's encoded instructions for a verb phrase switches the node's children so that the verb comes last. However, in some instances a more complex approach is required, and a node's children are reordered based on conditionals influenced by specific situations, instead of generalizations based on POS. For example, when a visitor visits an adjective phrase node, and it detects a "not" phrase attached to its specifier (left child), it reorders the node's children so that the "not" phrase comes after the adjective like it is in Japanese.

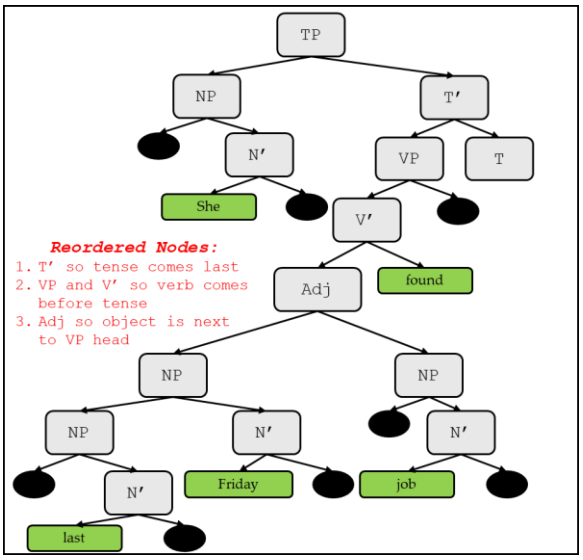

Figure 12: Node Reorder Visitor; In this example verbs need to come last in the Japanese sentence. Therefore, the verb phrase's (VP) specifier and bar node need to be reordered. Similarly, the verb bar (V') node's complement and head need to be reordered.

For "She found job last Friday," The `T′` node needs to be reordered so that its right child is the `T` terminal with tense information, which should come last. Then, both the `VP` and `V′` node's children need to be reordered so that the verb comes after the objects. Finally, the objects need to be reordered so that the direct object "job" is next to the verb "found." The reorder visitor will result in a sentence tree that reads "She last Friday job found."

(4) Word Translation Visitors

The node stripping and reordering visitor are similar in that they manipulate the vertex nodes of the tree. The remainder of the visitors manipulate the terminals of the sentence tree to achieve the grammatically correct representation of the Japanese syntax. This is done through (1) transferring English text to Japanese, (2) verifying or constructing Japanese conjugations, and (3) verifying or constructing Japanese particles. Since particles have a post-fixed position in relation to a word, they are ordered as the last task before achieving a translation. These

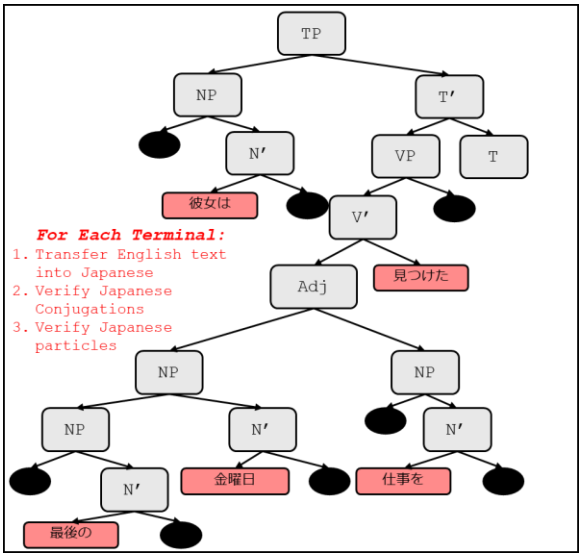

Figure 13: The resulting Japanese sentence tree after word translator visitors are finished processing.

terminals are also manipulated using a visitor pattern that relies on POS context to make decisions.

The translation visitor transfers English text to Japanese text upon visiting every terminal of the tree. At each terminal it calls a translate function that accesses an English and Japanese parallel corpus which maps an English word to a Japanese word (see Figure 14). This function is efficient due to the organization of lexical items in an English Japanese Parallel Corpus. A Parallel Corpus is one that maps the lemmas of the SL to the lemmas of the TL. Therefore, the word translation function relies on the efficiency of database software.

Parallel Corpus Data		
English Word	**Japanese Word**	**POS**
finished	終えた	Verb
slow	遅い	Adjective
earlier	先に	Adverb
morning	朝	Noun

Figure 14: English and Japanese Parallel Corpus that contains one-to-one mappings of English and Japanese lexical items and corresponding POS.

Unfortunately, since a custom parallel corpus was constructed for this project, this database pales in comparison to the previous English corpora databases. An existing well-formed parallel corpus of English and Japanese did not exist, and as a result, the project required the manual construction of a parallel corpus. The database is more concise and only includes words that are used in test cases. Inadvertently, this finite dataset might contribute to overfitting on test data—which could pose issues for future work.

The other two visitors are relatively complex. In the verification or construction of Japanese conjugations tense features and other modifiers such as negatives are searched and used as context to produce proper conjugations with the right tense and modality. In the example, the verb ("found") is conjugated correctly as *mitsuketa* (見つけた) through the parallel corpus data.

Since this visitor focuses on manipulating the terminals of the tree, in the Japanese representation words modified with "not" are still treated as different words. Using the syntax structure, "not" comes after a word due to the reordering visitor, and this matches with the Japanese construction of verbs. Japanese verbs are ordered with modifiable stem, a conjugation for tense, and a conjugation for its modality (positive or negative). Exploiting this fact, we conjugate verbs and adjectives in the sentence tree to include the stem and tense conjugation, but in the case of a negative "not" connected to its phrase, the tense conjugation is left out and handled by the "not" terminal. For example, "did not find" would be treated as two words "not" and "find." After reordering this phrase is "find not [past]," which the translation and conjugation visitors would translate to *mitsuke naka tta* (見つけなかった).

Simple conjugations for other POS exist and mainly pertain to the population of possessive *no* (の) particles. For possessive noun phrases like "teacher's instructions" this *no* particle is inserted between the two nouns. Identically, determiners and articles like "her wallet" inserts a *no* particle between them. In this example the constituent phrase "last Friday" uses an ordinal modifier, therefore we would need a *no* particle here as well. So, the translation of "last Friday," *saigo kinyoubi* (最後金曜日), would be modified to *saigo no kinyoubi* (最後の金曜日).

Conjugation also accounts for the diverse types of verbs and adjectives that exists with Japanese. Japanese verb conjugation also has three classes of verbs that require different conjugations. The *godan* (五段) verbs are the most common and has five possible stem conjugations. The *ichidan* (一段) verbs are common but have irregular conjugations due to an *-iru*/*-eru* (いる/える) verb ending. Finally, the irregular class of verbs contain the words "to do" (*suru* する) and "to come" (*kuru* くる). Adjectives have two classes: the adjectives ending in an *-i* (い) sound and adjectives ending in a *-na* (な) sound.

Finally, the algorithm ends with the particle population visitor. The verification or construction of Japanese particles is another complex algorithm which visits each constituent phrase's head and decides what particle is appropriate to attach for each phrase. In Japanese, each constituent phrase is assigned some particle which denotes its relationship with both the verb and the sentence. However, using X-bar theory we can discern the location of phrase specific particles. For example, the head of the TP's left child is the subject of the sentence [10]. Therefore, we can recognize "she" (彼女) as the subject and add the particle *wa* (は) to denote this function. Furthermore, the head of the right child's phrase in a V' is the object of the verb [10]. Similarly, we can recognize "job" (仕事) as the object and add the particle *wo* (を) to denote this function.

However, the particles for other constituent phrases like "last Friday" poses issues for the translator. In Japanese, most constituent phrases require a Japanese particle to give meaning to the phrase in relation to the verb. In the phrase "last Friday," we know that it is referring to a specific location in time, in which the particle *ni* (に) is used. However, such a decision cannot be made on POS context alone and will require some other module to effectively identify the correct particle to use.

After the verification or construction of Japanese particles, the translation algorithm attains a hierarchical tree representation of Japanese transferred text from English in which a traversal of the sentence tree would produce the correct sequence of lexical items.

(7) Results and Observations

The translator was able to produce Japanese translations for 60 English sentences in past, present, future tense, as well as their continuous, simple, and perfect forms. An excerpt from those test cases is provided in Figure 15 below.

Excerpt from Test Data			
ID	**English**	**Japanese**	**Tense**
1	I'll be giving a presentation at the conference.	私は会議でプレゼンテーションを供与している。	Future Continuous
2	I will not be attending the party tonight.	私は今夜にパーティーを出席していない。	Future Continuous
3	I will visit my parents next weekend.	私は次の週末に私の両を親訪ねる。	Future Simple
4	He will not pass the exam without studying.	彼は勉強しているせずに試験を合格しない。	Future Simple
5	She was singing loudly during the concert.	彼女はコンサートの間大声で歌っていた。	Past Continuous
6	I wasn't paying attention to the lecture.	私は講義に注意を払っていなかった。	Past Continuous
7	My friend and I didn't buy anything on our trip.	私の友達と私は私たちの旅行で何でも買わなかった。	Past Simple
8	Rachel didn't tell anyone the secret.	レイチェルは秘密を誰かに言わなかった。	Past Simple
9	I am currently working on a new project.	現在、私は新しいプロジェクトで取り組んでいる。	Present Continuous
10	We aren't going out for dinner this evening.	私たちはこれの晩夕食のため外に行っていない。	Present Continuous
11	He's learned to play the piano.	彼はピアノ遊ぶに習っている。	Present Perfect
12	We haven't seen that movie.	私たちはその映画を見なかった。	Present Perfect
13	My school starts at 8:00.	私の学校は8時で始まる。	Present Simple
14	We always eat dinner together.	いつも私たちは一緒に夕食を食べる。	Present Simple

Figure 15: Excerpts from the testing classes of the translator. Test were defined on the different types of tense features and their verb forms. The table provides two instances from each form of tense as provided in the "Tense" column.

The output of the translator gives insight to the type of sentences covered. In addition to all tense features, there are grammatical structures that are accurately represented in Japanese. For example, as seen in test #5 of Figure 15, the word "during" follows the Japanese pattern: Noun + の間. Furthermore, conjunctions in the context of two nouns, such as in test #7 of Figure 15, is accurately represented in the Japanese form: Noun + *to* (と) + Noun.

We also observe the correct nuanced ordering of POS in sentences like test #9 of Figure 15. In Japanese, adverbial phrases usually come at the beginning of the phrase, unlike English that usually introduces them after the subject. Owing to the X-bar structure, we can effectively separate "currently" from the subject "I" and reorder it so that *genzai* (現在), meaning currently, comes first in the sentence. The effectiveness of the reordering algorithm and X-bar structure repeats again in test #14 of Figure 15 where we effectively separate "always" from the subject "We" and reorder it so that *itsumo* (いつも), meaning currently, comes first in the sentence.

Smaller discrepancies are also observed to be implicitly solved by the translator. For example, in test #1 of Figure 15, the contraction “I’ll” and, in test #6 of Figure 15, the contraction "wasn’t” are handled by the preprocessor and converted into two separate words: “I will” and “was not,” respectively.

Despite the progress of the translator, there are also some problems that were encountered. First, the overfitting of the English-Japanese Parallel Corpus in the translation visitor handles cases that otherwise would not be handled by the translator. For example, in test #8 and test #13 of Figure 15, the words “Rachel” and “8:00” are examples of words that are translated well due to the overfitting of the parallel corpus. “Rachel” is a proper noun, and if a proper noun like “John” was used, the translation visitor would encode a null value into its terminal rather than *jon* (ジョン) in Japanese. Similarly, “8:00” is a concept of time, and the Japanese represent time differently. The parallel corpus returns “8 時,” but for other cases it would need to follow the format for Japanese time: Hour + *ji* (時), Minute + *bun* (分).

Furthermore, we observe that the accurate use of verbs is not being used effectively. For example, in test #11, the phrase “play the piano” does not use the correct word for “play” in Japanese. The current word it translates to *asobu* (遊ぶ) does not make grammatical sense in the context of a “piano,” and instead should be *hiku* (弾く). This error also occurs in test #10 where “going out” should be the verb *dekakeru* (出かける) and not *sotoni iku* (外に行く).

Lastly, we observe that Japanese particles are not populated effectively. As for translation, it is simple to translate prepositional particles due to their words in the parallel corpus matching the prepositional particles in Japanese. However, for other constituent phrases it was harder to discern the correct particle. This is because Japanese particles have nuanced applications that can not be discerned on POS context alone. Therefore, this translator assumes the particle *ni* (に) for certain noun constituent phrases, and the particle *wo* (を) for objects of the verb. While this assumption accurately portrays some particle usages, it is largely an ineffective feature that needs work.

(8) Conclusions and Future Works

The original intent of this project was the implementation of an SMT context-driven translator to provide accurate translations. While achieving the original intent, we also engaged in experimentation with SMT and X-bar theories—which are widely critiqued in their respective disciplines. In the context of translation, X-bar is a valid theory of phrase representation and was the representation that aided the translator in contextual decisions. These nuanced decisions are from X-Bar’s inclusion of a bar node, which helped separate and organize unique features of a word and permitted nuanced tree structures without compromising the sequential ordering of words. The theory was able to accurately represent all the test sentences in both English and Japanese, which is evidence of the translator’s utility and emphasizes its universality.

As for SMT, the context dependent decisions generated was able to completely build and reorder the sentence tree with minimal uses of the words meaning. Therefore, the data complexity of the translator

is optimized without needing to consider word meaning as often. However, that is not to say that SMT is effective in isolation. SMT in conjunction with NMT techniques can efficiently translate phrases with great accuracy. A concern for NMT is the potential overkill of a ML model as the sole function of a translator. SMT is effective in translating grammatical sentences because it can discern grammar rules and derive similar phrase representations in other languages. However, for the cases that were missed in the translator like particle population, word representation, and irregular uses of language can be solved through the training of different ML models focused on specific tasks of translation.

In particle population a future project would implement a ML model that tests on Japanese sentence data to classify what particle is appropriate for a constituent phrase. Japanese particles often have ambiguous or complex usages that may not reflect the POS present in a sentence. Therefore, this is an instance where NMT would bring this translator closer to producing natural translations.

Furthermore, idiom recognition and management are important components to have. Idioms are an irregular use of language, so therefore the translator would be able to provide an accurate phrase representation, but the resulting translation would not make sense literally or grammatically. Therefore, another ML model that makes associations between English and Japanese idioms would be beneficial in producing natural translations.

Similarly, word representation is another area for improvement. In the previous section we observed the misrepresentation of some verbs which were not used appropriately in the context of its object. This can be solved with word embeddings, which are a complex but useful representation of words that are able to exhibit strong relationships with other words with the same meaning. Such an approach would require a ML model that can make associations between words and meaning.

Last, as described in Section (4), the English-Japanese parallel corpus that stores English and corresponding equivalents in Japanese along with parts of speech was created as a bespoke utility. While not ideal, no other existing tool provided the nuanced information required by the context-based translation. As with any custom tool or database, this might result in overfitting of the parallel corpus to the test data. Therefore, extensive data acquisition and organization is needed to provide the possibility to translate more sentences.

The proposed extensions for future work will move toward correcting the mistakes by the translator. The culmination of these future works are the steps needed to take the translator from a proof of concept to a deployable software product.

(1) References

[1] M. Davies, "Word Frequency Data," *The Corpus of Contemporary American English (COCA),* 2008.

[2] R. H. R. J. J. V. a. G. B. Erich Gamma, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.

[3] A. Akbari, "An Overall Perspective of Machine Translation with Its Shortcomings,," *International Journal of Education and Literacy Studies,* vol. 2, no. (1), pp. 1-10, 2014.

[4] M. L. T. P. A. &. W. A. Dowling, "SMT versus NMT: Preliminary comparisons for Irish," 2018.

[5] M. D. Nimram, N. J. Azi, D. N. Nimram, B. S. Lagan, A. I. Umeh and O. & Nuhu, "Exploring X-Bar Syntax as Applied to English Syntactic Structures.," *Greener Journal of Languages and Literature Research,* 2023.

[6] R. Hausser, "Foundations of Computational Linguistics," in *Man-Machine Communication in Natural Language*, Erlangen-Nürnberg, 1994.

[7] J. Hirschberg and C. D. Manning, "Advances in Natural Language Processing," *Science, 349(6245),* pp. 261-266, 2015.

[8] R. M. Losee, "Natural Language Processing in Support of Decision-Making: Phrases and Part-of-Speech Tagging," *Information Processing & Management 37, no. 6,* pp. 769-787, 2001.

[9] H. Phan and A. Jannesari, "Statistical Machine Translation Outperforms Neural Machine Translation in Software Engineering: Why and How," *Association for Computing Machinery,* p. 3–12, 2020.

[10] C. Anderson, B. Bjorkman and D. Derek, "Essentials of Linguistics," 2nd ed., eCampusOntario, 2022.

[11] J. Palsberg and C. B. Jay, "The Essence of the Visitor Pattern," *The Twenty-Second Annual International Computer Software and Applications Conference (Compsac'98) (Cat. No. 98CB 36241),* pp. 9-15, 1998.