PARSING CODE-SWITCHED TAGLISH LANGUAGE BY CREATING CONSTITUENTS

by

Fadiah Qudah

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science at
The University of Texas at Arlington
December, 2019

Arlington, Texas

Supervising Committee:

Vassilis Athitsos, Supervising Professor
Chris Conly
Shawn Gieser

ACKNOWLEDGEMENTS

First of all, I thank my advisor Dr. Vassilis Athitsos for providing an environment that truly allows students to pursue their interests. His guidance style fosters academic growth to those who truly seek it while simultaneously grounding students when they (inevitably) veer off course. His extraordinarily high expectations coupled with his utmost patience demand constant improvement, and for that I am extremely grateful. My advancement in this field is largely due his support and I consider his direction as an essential component of my ongoing development.

I would also like to thank Dr. Bahram Khalili for his initial help when I made the switch to computer science and his continuing encouragement since then. His seemingly unending positive attitude encourages students to continue moving forward, no matter what circumstances occurs.

ABSTRACT

PARSING CODE-SWITCHED TAGLISH LANGUAGE BY CREATING CONSTITUENTS

Fadiah Qudah, M.S.

The University of Texas at Arlington, 2019

Supervising Professor: Vassilis Athitsos

When extracting meaning from language, a common first step is to break down language into constituents, or words that work together as a unit. This task, known as parsing, typically follows a specific grammar in order decompose the language into its underlying structure composed of constituents. Difficulties with this grammar-based parsing occur, however, with real-world natural language due to its unstructured nature. Code-switching, the phenomenon of alternating between languages while communicating, further complicates this task by requiring us to parse based on two (or more) languages instead of one. In this thesis, a data-driven method to parse code-switched language into its constituents is presented. The code-switched language used in this thesis is Taglish, comprised of English and Tagalog, and the data is collected from the social media site Twitter.

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

With over half of the world's population speaking at least two languages, code-switching situations are ubiquitous. The phenomenon of code-switching[1], "the shift from one language to another [1]", occurs in various multilingual societies for a variety of reasons and motivations. This occurrence is generally considered to occur in more informal scenarios, such as in movies and texting with friends, but this observation is far from a steadfast rule. Different communities may use varying levels of code-switching and the ever-evolving nature of language and language usage results in a dynamic switching process. This global occurrence of code-switching has had the unintended, yet undeniable, consequence of adding more complexity to an already complicated natural language processing arena. One specific issue is posed in an initial step of natural language processing: natural language representation in the computer.

Code-switching example between English and Tagalog:

1. **Well-acted** lahat ng **cast. Thank you.**
   Well-acted by the whole cast. Thank you

Figure 1.1 An example of codeswitching. Two languages, English and Tagalog, are used in the same utterance.

From a surface perspective, natural language processing can be viewed as two main tasks: natural language understanding and natural language generation. Natural language understanding is the task of the machine understanding natural language (as a human would understand) and natural language generation is the task of the computer producing language to convey some information (as a human would). This process can be thought of as a replication of actual human interaction, where a person can understand language and generate language. Very often, during a flow of communication, person A generates language that encodes some thought and person B understands the message that was encoded by person A. With what person B understands, person B might then do some action, such as analyze or generate language back to person A. This concept is summarized in figure 1.2 below.

[1] Note there is traditionally a distinction made between code-switching and code-mixing. In this thesis, the two terms are used interchangeably to denote the switching from one language to another in communication.

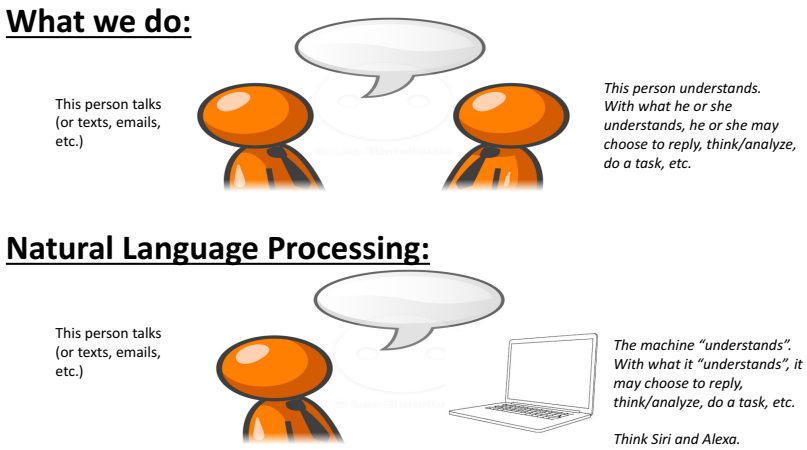


Figure 1.2 Top: When people communicate, a person can generate language (left) and a person can understand language (right). Bottom: When a person communicates, the machine can "understand" what is communicated and can generate a natural language response

Natural language representation is a component in natural language understanding. It consists of taking the natural language data and extracting an intended meaning. This meaning is then stored in some computer friendly representation and serves as an "understanding" from the original natural language data. From this "understanding," the computer can then complete downstream tasks, such as translation or natural language generation. For example, in order for a computer to translate a French sentence to English, it must first "understand" the French sentence by holding a representation of the intended meaning of this French sentence, as shown in figure 1.3. This representation should hold as precise of a meaning as possible, because the computer's attempt to translate this French sentence relies solely on this representation. It is the same way with a human; if a human did not accurately understand the starting English sentence, the resulting translation will not be accurate. The end result, a generated sentence in English, will only be as accurate as the representation that it started with.

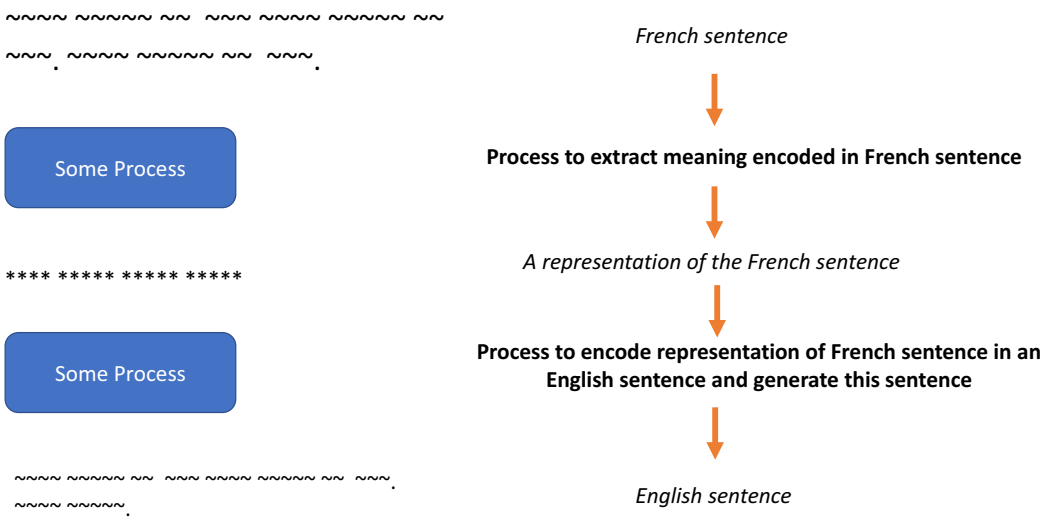


Figure 1.3 A general overview of the steps taken to translate a sentence from one language to another

So how does this process of extracting meaning from natural language data and encoding it in a computer-friendly representation work? While many methods exist, a common first step is to attempt to pull the meaning out from a group of words that act as a unit rather than from each individual word. Groups of words that act together as units are called constituents. The process to realize these constituents is called parsing and can be defined as "the process of structuring a linear representation in accordance to a given grammar [2]." The grammar used is the set of rules that dictate what constitutes an acceptable underlying structure from a language and outlines constituent structure and eventual constituent combination to create acceptable language. This process is shown in figure 1.4.

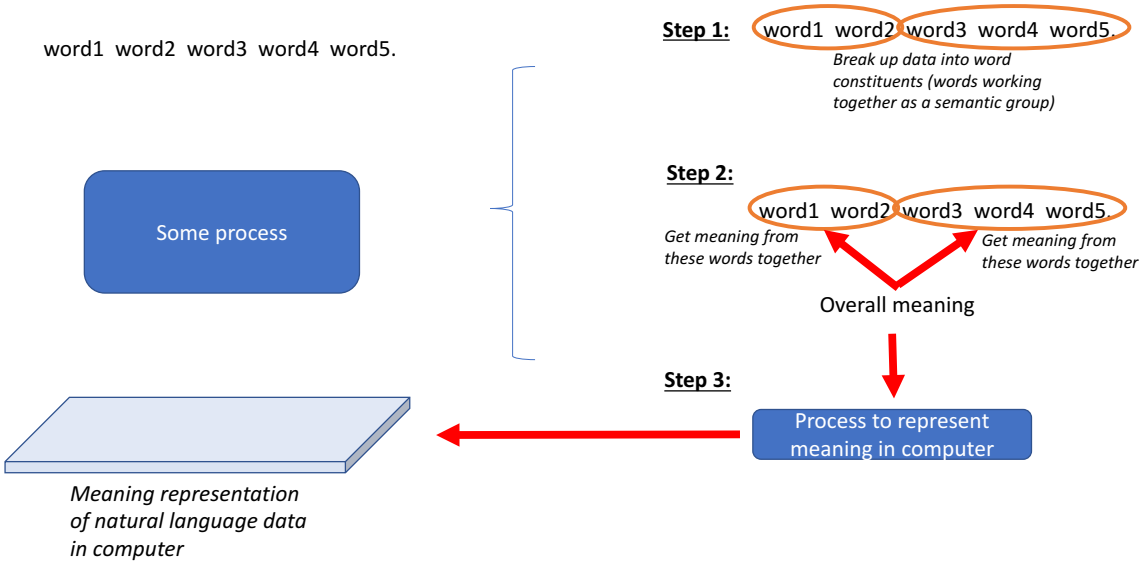


Figure 1.4 The overview of representing meaning by first breaking up natural language data into constituents. Step 1: break the word up into constituents (groupings of words that act together as a unit), Step 2: Get meaning, Step 3: Represent the overall meaning extracted in the computer or machine

So how can we start parsing and breaking up this natural language data into constituents? A possible first step is to break the data up into sentences (indicated by an ending punctuation like a period) and then further break each sentence down into acceptable constituents based on the grammar of the language. In order to successfully do this, we first label words by their part of speech, or a way to categorize each word by its syntactic function. The grammar then gives the rules on how to acceptably combine these words based on their part of speech and create appropriately formed constituents. These constituents can be further combined or nested until reaching a final, larger constituent. This final constituent represents the final parse of the sentence. One caveat to this grammar-based parsing, however, is a prominent and inherent feature in natural language: the fact that natural language data does not always strictly follow the rules of grammar. This fact must then be accounted for in order to parse real-world language data. In this case, the definition of parsing by Nivre might be considered more general and fitting: “The automatic analysis of syntactic structure [3]”.

Code-switched natural language data brings an additional consideration to this parsing process. Code-switched language alternates between different languages at various points, meaning that the composition of constituents can differ depending on the language being used. For example, in the codeswitched sentence used in the beginning of this section (see figure 1.1), the underlying structure is comprised of two languages: Tagalog and English. Since our eventual goal of extracting meaning relies on constituent detection in parsing, the unstructured nature of natural language coupled with the grammatical fluidity of code-switching poses a problem for us. In this thesis, a data-driven statistical method to parse code-switched data (one sentence at a time) is used to output the parsed version of the sentence. The parsed version of the sentence is represented by the constituents of the data with surrounding brackets (see figure 1.5).

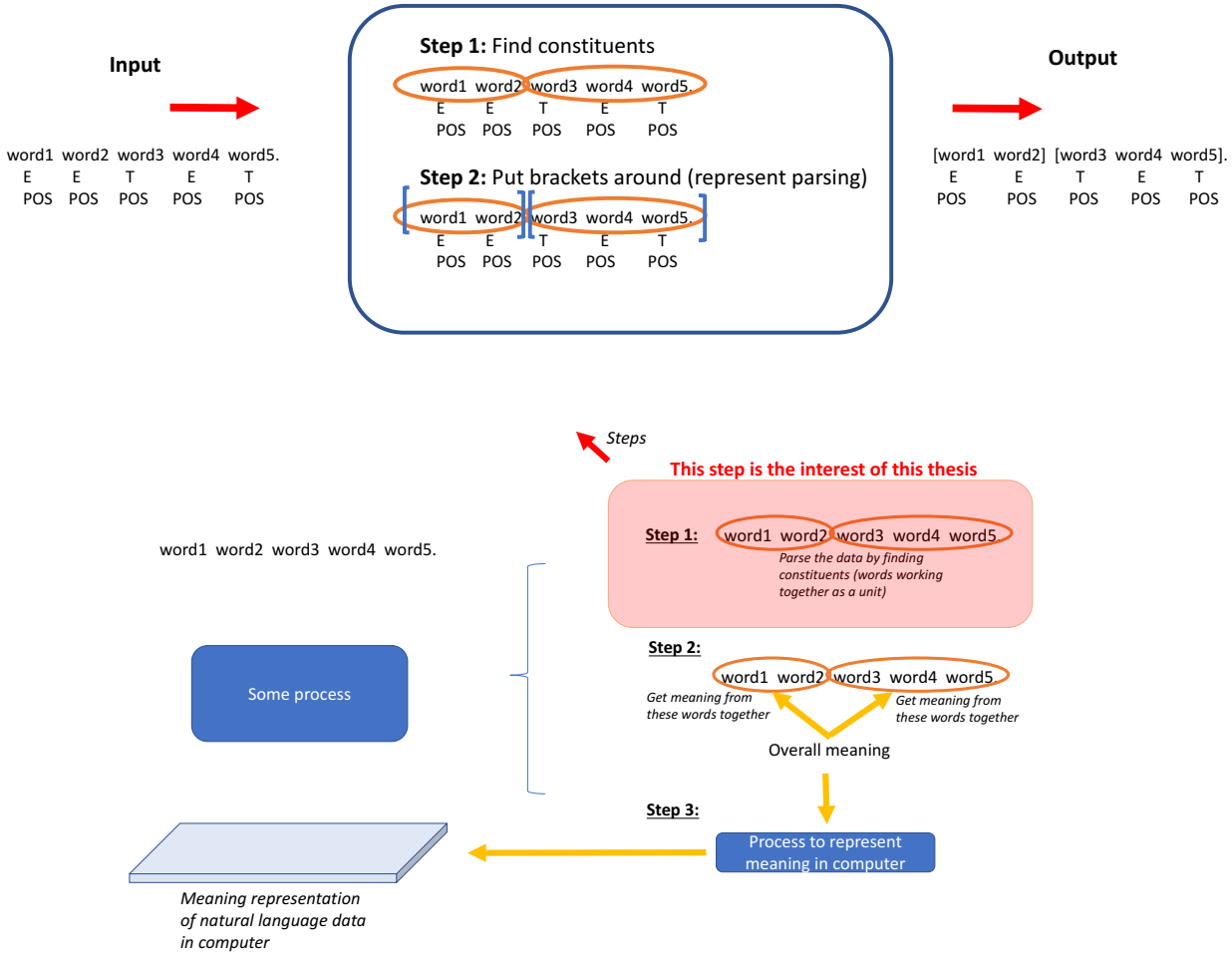


Figure 1.5 Bottom: the overall idea to obtain a meaning representation on the computer. Top: the focus of this thesis.

For this task, one specific case of code-switching called Taglish is considered. Taglish, a combination of English and Tagalog[2], occurs in Filipino communities and can frequently be seen and heard in daily Filipino life. It is not uncommon to hear Taglish used in movies or with family and friends in Filipino communities. Below, I go over a bit of information about code-switching and parsing in order to introduce the reader to the main components of this thesis.

1.1 Code-switching

Code-switching is the act of alternating between languages while communicating and should not be confused with word-borrowing, which is when a word from another language is adapted phonologically or morphologically to the language. It should also not be confused with scenarios that use vocabulary from one language and simply plug it into another language's grammar structure. When code-switching, words do not suffer from syntactic or phonological alterations [4]; it is truly a combination of languages [5]. It can occur along the whole spectrum of society, in both formal to informal settings.

We can see code-switching occurring at all levels in communication: from the smallest unit of meaning, the morpheme, to the sentence level and topic level. For example, a morpheme indicating plural, like–*ler* in Turkish, can be added to the German word *Aufgabe* to result in a single word composed of both languages: *Aufgabeler*. This is an example of an intra-word switch. Together, -*ler* and *Aufgabe* combine to give one meaning-a plural value of *Aufgabe.* Additionally, switching can occur between words, clauses, sentences and even whole conversation topics.

While code-switching can be seemingly random at first, various frameworks exist to describe it. A well-known existing model is Carol Myers-Scotton's Matrix Language Frame model (MLF), which assigns languages used in code-switching as either a Matrix Language or Embedded Language. Fundamentally, this model claims that the languages do not play equal roles in code-switching; the Matrix Language is considered to be the dominant language while the Embedded Language plays a more restrained role. Additionally, the surface morpheme order and the grammatical frames are supplied by the Matrix language [6]. This means that outwardly, the structural appearance of the Matrix Language may be easier to discern that that of the Embedded Language.

[2] The national language of the Philippines is Filipino, a standardized register of Tagalog

The Matrix Language can possibly, but not definitively, be determined by three different components. The first component, a psycholinguistic component, claims that the speaker's first language operates as the Matrix Language in a majority of instances. The second component, a sociolinguistic component, claims that the community's dominant language in turn becomes the speaker's Matrix Language. The third component claims that the Matrix Language can be assumed by the majority of morphemes used. Another well-known model is from David Sankoff and Shana Poplack's 1981 paper "A Formal Grammar for Code-Switching [7]". This model looks at code-switching through the constraints on language combination.

Frequent travels and visits around the world show us that unilingualism is not the norm; a majority of the world is bilingual and even multilingual [8]. This constant exposure and interaction with a variety of languages inevitably fosters the use of more than one language for any given situation. More access to language resources gives a communicator the ability to express what otherwise may not be possible. For example, a speaker could use a word that encodes a specific meaning that may be present in one language and not the other. Another example may be when one cannot find the word in one language due to lack of fluency. Bautistia [1] distinguishes between two types of code-switching: proficiency-driven and deficiency-driven. The former occurs when the person is fluent in the languages used and switches "for maximum efficiency or effect". Whether or not this is an active decision is not always obvious; various evidence exists that, barring conscious efforts to fit in with a specific audience, people frequently code-switch subconsciously [9]. The latter is when a person is driven to switch due to lack of language skill in the current language in use.

It is often documented that the reason for code-switching cannot just be reduced to the fact of switching languages for communication alone. As mentioned by Gal, "codeswitching is a conversational strategy used to establish, cross or destroy group boundaries; to create, evoke or change interpersonal relations with their rights and obligations [10]." Indeed, from a socio-linguistics standpoint, one can actively or subconsciously choose to strategically code-switch in order show or distance one's self from a social group.

1.2 Taglish

Taglish is a code-switching variety that uses English and Tagalog. Tagalog is a language spoken in the Philippines, and Filipino, a standardized register of Tagalog[3], is a national language of the Philippines. English is also designated as a national language and is currently the dominant medium of instruction in schooling. The country is composed of some 170 various languages and mutually-unintelligible dialects [11], allowing for numerous occasions for code-switching to occur.

Taglish Examples:

1. **Well-acted** lahat ng **cast. Thank you.**
   Well-acted by the whole cast. Thank you

2. **I really like Julia**. Nakaka-**miss** talaga.
   I really like Julia. I really miss [her].

The relationship with Philippine languages and foreign languages are marked and influenced with the country's colonial past. Even the country's name *The Philippines* comes from the Spanish King Philip II, who was in power during the 16th century colonization of the islands [12]. The Philippines was under Spanish rule for 333 years and from a linguistic standpoint experienced a significant impact during this time. Long-standing remnants of the effects of this Spanish rule on language can still be seen to this day, particularly on the Romanized script of Philippine languages and the presence of Chabacano, a creole considered to be "the most extensive Spanish-based Creole language now in existence, and the only one found outside of the Americas [13]". American influence on the Philippines began after the Spanish-American war of 1898 which resulted in the Philippines becoming an American colony until 1946. During this colonial time period, Filipinos became familiar with American culture and English, unwittingly helping to set the scene for code-switching between Tagalog, at that time already designated as the national language by President Quezon, and English to take place.

[3] Note that Tagalog can also refer to the Tagalog people. In this context, I am referring to the Tagalog language originally associated with the Tagalog people. It was given the name Pilipino, or Filipino as stated by the 1987 constitution, to designate it as a national language instead of an ethnic language.

Currently, both English and Tagalog as standalone languages have their typical roles in Philippine society. Professional domains, such as medicine and commerce, largely use English and Filipino children are exposed to English in formal classes during early years of their schooling. It is noteworthy that much like other places in the world that use English, a standardized form of English, called Philippine English, has emerged. Tagalog, in its standardized register of Filipino, is spoken by at least 84% of the population [14] and "serves as the language of commercially driven mass media, specifically radio, television, and film [15]."

Several linguistic studies involving Taglish exist. An early examination of Taglish was done in a 1967 thesis by Azores [16], followed by several other theses and papers dealing with Taglish. "The Filipino Bilingual's Competence: A Model Based on an Analysis of Taglish-English Code Switching" is a thesis by a well-known Taglish scholar named Maria Lourdes S. Bautista. Her analysis "would eventually be given concrete labels by Myers-Scottson [6] and Poplack and Sankoff [8],[5]." For example, Bautista speaks of a "base language" in her analysis; this appears to be the dominant Matrix Language later used in Myers-Scottson's Matrix Language Frame Model.

It is worth noting that attempts to process Taglish computationally are much sparser; at the time of writing this thesis, NLP and computational research dealing with Taglish alone were difficult to come by. Various studies dealing with Tagalog exist, such as the dependency parsing work done by Manguilimotan and Matsumoto [17] and an attempt to build a machine translation system for English-Filipino by Roxas et al. [18].

1.3 Parsing

As mentioned in the introduction, the importance of parsing natural language data in the field of natural language processing lies in the need to extract encoded meaning from the data. Once this preprocessing step is complete and we have the meaning, we can represent it in the computer. While early starts in natural language processing looked for the meaning of single words, it was soon realized that meaning lies in combinations of words (and sometimes whole sentences). Single words in isolation would not truly give the whole meaning.

According to Grune-Jacobs, parsing can be more generally described as "the process of structuring a linear representation in accordance with a given grammar [2]." This definition of course presupposes that a grammar for a language is already defined, which is not always the case. Additionally, even a language with a defined grammar may be produced ungrammatically in real world instances. For that reason, Nivre's more general definition may be better suited: "The automatic analysis of syntactic structure [3]."

Parsing can be classified in different ways. A few of these distinctions are discussed below.

1.3.1 Grammar-Driven versus Data-Driven Parsing

In both grammar-driven and data-driven parsing, the desired outcome is parsed natural language. What distinguishes the two methods is how this outcome is achieved: grammar-driven techniques rely on a grammar while data-driven techniques rely on the data. Grammar-driven parsing parses by using the grammar of a language. For example, in English we have specific grammar rules that dictate how words can be combined. We cannot, for example, place a determiner after a noun:

1. **The** dog eats. ✓
2. Dog **the** eats. x

In the first sentence, the determiner, *the*, precedes the noun *dog*. This is an acceptable combination according to English grammar. In the second sentence, we see the grammatically unacceptable sequence of the noun followed by the determiner. It is these rules that grammar-driven parsing uses to parse language.

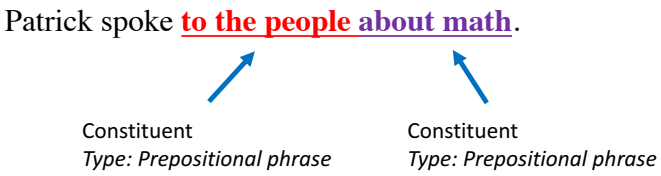
Data-driven parsing, on the other hand, instead uses different techniques to discover an underlying pattern of the syntactic nature of the data. These techniques can vary, but commonly used techniques are statistical modeling and machine learning techniques [19]. One issue with data-driven parsing is the same issue encountered with other data-driven tasks: the amount of data required to train and give a good performance. Code-switched language, especially when composed of one or more low resource languages, proves difficult to find enough data for. As stated by Nivre et al. "approaches based on completely unsupervised learning are still vastly inferior in terms of accuracy, there is consequently a need for supervised approaches that are resilient against data sparseness [20]."

From a practical standpoint, some combination of grammar and data-driven techniques can be used to achieve a successful parse.

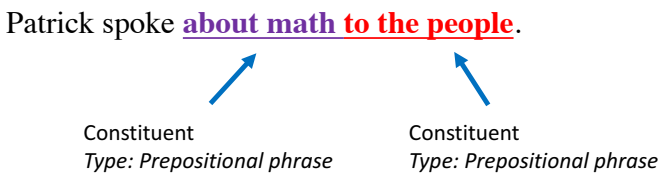
1.3.2 Dependency versus Constituency Parsing

Dependency parsing works by establishing relationships between "head" words and words that modify those heads [21]. Constituent parsing, on the other hand, abstracts one level up from the actual words that make up the natural data by viewing groups of words as constituents, or groups of words that act together as single unit based on their different syntactic categories [22]. The difference between these two can be summarized in the following way: dependency parsing looks to break up data based on relationships while constituent parsing breaks up the data into sub-phrases.

To prove that a group of words form constituent, a test for constituency can be applied. For example, take the sentence *Patrick spoke to the people about math*. Let's take a look at two constituents found in this sentence, specifically of the type prepositional phrase:



According to linguists, one piece of evidence point to the existence of the prepositional phrase constituent is that we can rewrite the same sentence, where we can move around the constituents as one unit:



Another example of evidence is that we can move a constituent to the front (*To the people, Patrick explained math.)* among other things.

Constituents, in addition to being next to each other, can also nest within one another. We can visually see constituents in bracket form (where each word is marked with a syntactic category and then combined together to form a constituent. That constituent can then become part of another constituent etc.):

```
(S (NP Patrick)
   (VP spoke
       (PP to
           (NP the people))
       (PP about
           (NP math)))
   .)
```

The same parsing above can be expressed in a linear fashion:

```
(S (NP Patrick)(VP spoke(PP to(NP the people))(PP about(NP math))).)
```

A parse can also be represented as a parse tree (S means sentence, VP means verb phrase, PP means prepositional phrase):

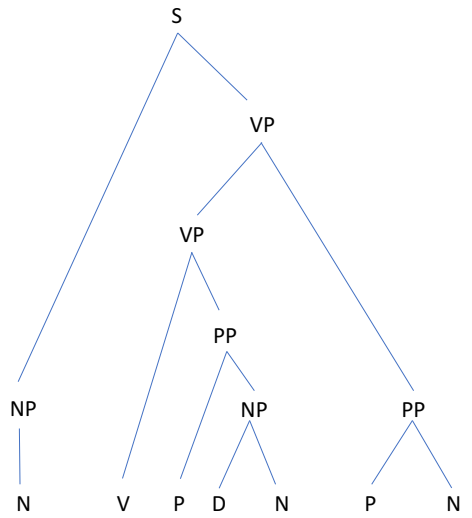


Figure 1.6 An example of a parse tree

1.4 Objectives and Research Questions

The objective of this thesis is to take code-switched natural language data and output parsed code-switched natural language data. Parsing will be represented by identifying constituents of words (groups of words that act together as a unit) and placing brackets around them (see figure 1.7). The constituents will be identified through a data-driven approach rather than a grammar-based approach.

Figure 1.7 The input will be transformed into an output with brackets around the constituents.

Some of the main research questions that support this endeavor are:

1) How can we successfully parse natural language data without a specific grammar? In this case, how can we rely strictly on the data to find an underlying syntactic structure? How can we apply this concept to code-switched data, which has its own underlying structure?

2) Can we use existing monolingual data-driven parsing techniques for the languages that are used interchangeably in Taglish (English and Tagalog)? If so, in what way?

3) What would be a good source of Taglish data? Since code-switching in Taglish tends to happen in more casual settings, where can we find casual data? Would audio or text data be a better choice?

1.5 Outline of chapters

Chapter two will go over related work in parsing and code-switched language processing. Specifically, we will look parsing and code-switched language processing before looking specifically at code-switched parsing related work. Chapter three will go over the data and method. Chapter four will discuss the results and look at future directions.

2. Background and Related Work

In the previous chapter, I discussed the various components of this thesis along with their inherent characteristics that affected the process of parsing code-switched data. Since code-switched data can contain two or more grammars, successful parsing of the data relies on recognition of this fact. Some studies treat code-switched language as a single language itself, even while using information for each individual language to generate information. Considering that some linguistic theories posit the use of a third grammar when dealing with bilingual codeswitched language [23], this is not such a radical idea. If we consider our code-switched data not as a jumble or mix of multiple languages but as a language itself, we can significantly reduce our parsing workload. This reduction would come from the fact that we are only looking to explain a single structure and implement parsing rules on that single language. This does not mean we cannot use the single language resources available to us in various steps. We are simply abstracting to a single viewpoint when it comes to the actual processing. For example, inter-sentential code-switches can benefit from monolingual data due to the fact that complete sentences are in one language. This fact would allow us to use that specific language's grammar to parse that specific sentence.

It is important to note that while viewing code-switched data as its own single language has its positives, additional considerations follow. For example, by viewing code-switched data as a single language, we acknowledge we are dealing with less data when processing. This is due to the fact that most code-switching situations are informal, and often not documented, scenarios. While Taglish data itself is not by any means impossible to find, the amount of data is far less than English or Tagalog standalone data, where Tagalog is already considered a low-resource language[4]. This would classify Taglish as a low resource language itself, meaning we have significantly less data to work and draw conclusions from. Additionally, the Taglish data that is available would most likely not be as well annotated as its monolingual components.

Below, I first go over related work in parsing and code-switched language processing. I then take a look at related work in parsing code-switched language.

[4] A low resource language means the language does not have a plethora of data to work from and thus potentially limits discoveries and subsequent results that can come from that data.

2.1 Related work in parsing

Parsing has advanced substantially over the two decades [23]. Before looking at related work that deals directly with parsing code-switched data, we first look at general parsing techniques and considerations on monolingual data. Historically, more effort has been spent processing monolingual data. Additionally, a substantial amount of research dealing with parsed code-switched data starts with examining techniques used on monolingual data. These techniques are then adapted or modified to fit code-switched data needs. Much of the related work in this section pertains to parsing with regards to the languages that compose Taglish: Tagalog and English.

One factor that heavily contributed to parsing, as well as other processing tasks, was annotated datasets. The use of collected natural language data in the 1990s gave researchers a chance to view language tasks, such as parsing, through a large bank of data. In 1993, Marcus et al. introduced the *Wall Street Journal* Penn Treebank, part of an 8-year language collection and annotation task, to the world [24]. This sizable English corpus, collected between 1989-1996, included around 7 million words of part of speech tagged text amongst other items of interest. It provided the base for many researchers to apply statistical parsing to a large-scale bank of natural language data. The results of the work done by these researchers were extremely productive; statistical analysis on English parsing improved dramatically over the years. These outcomes inevitably encouraged the development of numerous new banks with annotated data for additional languages such as Arabic, French and Swedish [19].

When parsing, inherent language feature considerations impact results. Tagalog, one of the languages that makes up Taglish, is considered a morphologically rich language while English, the other language that makes up Taglish, is not considered morphologically rich. A morphologically rich language means that word level information is considered important when giving meaning [17]. Tsarfarty et al. describe the three main challenges faced with parsing morphologically rich languages: the architectural challenge, the modeling challenge and the lexical challenge [19]. English does not have encounter this; it is more about word placement and order to give meaning. Since I am choosing to treat Taglish as a single language (composed of the two syntactic structures and grammars of the languages used), it will inevitably have a

morphologically rich component to it and thus a system processing Taglish should be aware of this characteristic.

Another feature that affects parsing would be word order. A striking difference between English and Tagalog is the rather free word order exhibited by Tagalog [25]. A free-order word language means that the word order is not strict. The relationship between morphologically rich languages and a more relaxed word order is noted in research, so it should not be surprising that a morphologically rich language like Tagalog has a freer word order [26]. The reason given for this correlation is simply that if we rely on attaching meaning to words, word order no longer becomes important. For example, Latin and German both have a freer word order than English for example and they have a more morphologically rich aspect.

Finally, as previously mentioned, one fact about natural language that adds complexity is the fact that our data does not always conform to strict grammar constraints for a language. For example, when someone answers a question, they may not answer in a complete sentence:

Q: Where should we throw the party?
A: At the school.

At the school is not a complete sentence (it is a part of a sentence), and a parse tree for the answer would look something like the following:

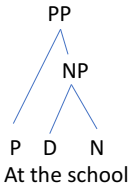


Figure 2.1 A parse tree

This is a preposition phrase which is not meant to stand alone as a sentence. Another example might be someone simply replying *School*, which is a perfectly acceptable answer when using natural language, but once again not fitting to the constraints of a complete sentence by English grammar standards.

We can see this occurring in a Taglish example where we end up with a NP rather than a complete sentence:

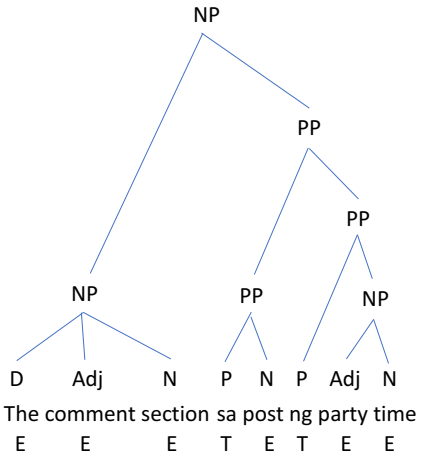


Figure 2.2 A parse tree with Taglish

Since the goal of natural language processing is to process actual natural language, we have no choice but to somehow account of this issue [27]. It should be our goal to have our system fit natural language and not to force natural language to fit our system. An initial approach to this occurrence was to simply identify low-level constituents[5]. For example, in our example above, even though it is not a complete sentence it can be still be described as a prepositional phrase; this low-level constituent, while not a complete sentence, can still describe our data.

2.2 Related work in code-switched language processing

In this section, we take a look at related work in code-switched language processing. Specifically, we take a look at tasks used in this thesis such as language identification and part of speech. This is for the same reason as looking at related work in monolingual parsing: since much less research exists on the topic, any work done can be useful and potentially helpful in parsing code-switched data. The fact that in 2008, Solario [28] claimed that even something as simple as little to no part of speech tagging and parsing research existed on code-switched data (as opposed to with monolingual data)[6] shows the relatively little work done on the field of parsing and part of speech of tagging in codeswitched language.

[5] the "lower nodes" of the parse tree, not going to the root (sentence)
[6] At that point, there were theoretical approaches, but no implementations or work done on actual corpora

A commonly used task is language identification. When dealing with monolingual input, language identification has pretty good performance [29] and is considered to be a solved problem. This is partially due to a substantial presence of monolingual annotated data [23]. One issue does occur, however, with monolingual data that have significant overlap. In work done by Tan et al., we can see this issue Croatian and Serbian [30]. This is also true in code-switched languages that have similar structures and dealing with code-switching vs borrowing.

Another task is part of speech tagging. This task assigns a category for each word regarding its part of speech and is a common pre-processing step when dealing with parsing. A huge issue with general part of speech tagging is a lack of annotated data, further exacerbated by data that is code-switched. It would appear from the work done by Solorio and Liu [28] that using monolingual taggers alone do not work well with code-switched text. Three common techniques employed to overcome this issue are: using monolingual tagger outputs based on probabilities, utilizing monolingual language resources and applying machine learning on annotated CS data [23].

2.3 Related work in parsing code-switched language

Related work in parsing code-switched language is unfortunately not as prevalent as its monolingual counterparts. As mentioned in the previous sections, data availability, particularly annotated data, plays a role in parsing and processing language. Due to the lack of code-switching data available, parsing code-switched language has been difficult. With parsing alone on monolingual data, especially with the rise of deep learning, a lot of parsers have been developed. Unfortunately, this is not the case with code-switched data. An additional issue lies in the language structure itself: a mix of different structures from different languages. We can see this example when a sentence follows the syntax of one language for most, but not all, of the sentence. Other examples are when lexical items of one language follow the syntax of another language or one syntax interferes with another.

Parsing theories on code-switched data were first mentioned in the early 1980s [31], but actual code-switched parsing itself is limited. One study with Sharma et al. contributed a corpus with language IDs, normalization, chunk boundaries and labels [32]. It dealt with Hindi-English social media text and had a

chunking accuracy of 61.95%. Another study with Vilares et al. had an encouraging experiment that merged the training sets of two languages [33].

3. Data, Method and Results

In this chapter, we introduce the Taglish dataset amassed through Twitter and the motivation of using Twitter as a data source. We then introduce the method and discuss the results.

3.1 Data

Taglish code-switching tends to occur in more informal scenarios, such as during casual conversation or daily TV shows. In formal atmospheres, such as official government announcements, dialog tends to be completely in one language**.** Additionally, a person tends to alternate between languages less when made explicitly aware for monitoring of code-switching**.** Because of these factors, a specific source of casual Taglish data made without monitoring was required. Many different sources, such as email and text messages, were considered before finally arriving at text-based social media posts. A text-based social media post is where a user writes out a message and posts it to a platform such as Facebook, Twitter and Instagram. Using social media posts fulfilled the two criteria mentioned above: a casual atmosphere not focusing on code-switching production. After examining all three of the social media sites mentioned above, Twitter was ultimately selected. On Twitter, users can post natural language text in the form of tweets or comments; these tweets and comments can then be used for analysis.

On Twitter, there were two main items of interest for data collection. These two items were the original tweets created by people and the comments posted in response to these original tweets. After registering for a Twitter developer account, a program in Python was developed (using the API for Twitter) to scrape for tweets. To begin, a preliminary search for common Filipino pop culture topics was carried out. This search started by looking for tweets about celebrities and sporting events, as these topics typically foster natural, casual conversation. From this initial search, tweets from related accounts were continuously scraped. At the end, these Twitter scrapes yielded 19,789 total tweets[7]. After going through the tweets, this number was reduced to 15,993 total tweets exhibiting some sort of code-switching between English and Tagalog. The tweets were further filtered down to remove inter-sentential switches (since a complete sentence in a single language would not need special parsing considerations techniques) and intra-word code switches (since parsing would be by syntactic category and no lower than the word level), leaving us with a total of 11,981 tweets to use.

[7] No identifying user information was retained-just the actual text from the tweet or comment.

3.1.1 Text normalization

Text normalization is the task of standardizing text that deviates from some agreed-upon (or canonical) form [23]. When dealing with social media data, defining a standard can be difficult due to the unstructured nature of casual natural language. Ultimately, three human judges deemed what was considered a natural and straightforward form. A two-step process was then taken to clean up the text. The first phase, dealing with superficially cleaning the text, consisted of handling:

1. Excessive and overzealous use of punctuation (for example, using ….!… instead of a single period to separate clauses and sentences)

2. Sporadic punctuation (random periods, dashes and commas). Data was manually annotated for the end of a sentence with a period if a period was not included.

3. Actual punctuation. Since we cannot count on natural language data being appropriately punctuated, all punctuation (except for periods indicating the end of a sentence) was removed.

4. Smiley faces, emojis, emoticons

5. Typos, random letters

It is important to note that all these adjustments might have relevance in other types of natural language processing work, even though they were hindrances in this thesis. For example, smiley faces and emojis could possibly add some meaning into the data, as shown below:

1. I am 😀.

2. I am 😀 😀 😀.

3. I am 🙂.

In the three examples given, the emojis (type and number) can be used to indicate different meanings that could be encoded into different sequences of words. For example, it is quite possible for the 😀 😀 😀 sequence to indicate *very, very happy* while the single 😀 means just *very happy* and the 🙂 meaning *I am happy*. In our case, if a human judge decided that removing the emoji appeared to completely change the meaning of a sentence, the data was removed from our dataset.

After the initial cleaning step, the next phase was to address the following for meaning in the data:

1. Words that express meaning but are not necessarily considered to belong one language, such as *hahahahaha*. These cases posed difficulty since they do not have a specific syntactic category or expected location in the sequence of word data.

2. Extreme slang or exaggeration. For example, words like *really* were sometimes written as *reallyyyyyyy* to convey different meanings. *reallyyyyyyyyy?* would have a different meaning from *I am reallyyyyyy tired*. The first *reallyyyyyyy* would indicate surprise or disbelief about a subject while the second *reallyyyyyyy* is used as a stronger modifier for *tired*.

3. Handling Ligatures. A ligature is a particle that connects modifiers (such as adjectives) and the words being modified (such as a noun). In Tagalog, ligatures can either exist as either a stand-alone particle (*na*) or as part of the preceding word (for example, *-ng* if the preceding word ends with a vowel). For our purposes, ligatures that were connected to a preceding word were separated, allow us to count ligatures like *-ng*.

After this two-step cleaning process, the next phase was to annotate and prepare the data for input into the parsing system.

3.1.2 Language Annotation

The data was first manually annotated for syntactic category and language by three human annotators. The three human annotators were fluent in both English and Tagalog as well as frequent users of Taglish.

Data	**Good**	**job**	**lahat**	**ng**	**cast**
Language	E	E	T	T	E
Part of Speech (POS)	POS-adj	POS-N	POS-adj	POS-P	POS-N

Figure 3.1 Annotated code-switched data

Each word was first annotated for language (E for English, T for Tagalog). As mentioned in the related work, difficulty when annotating code-switched language can arise with language identification and the human annotators were not immune to this. Specifically, the annotators had limited difficulty differentiating between words that might have been borrowed from one language to another (in most cases, English words borrowed into Tagalog) and an actual code-switch. In these cases, they resorted to looking up the word etymology and history to make a final decision on the language classification.

Words were then annotated for parts of speech. The parts of speech used were:

1. n-*Noun*
2. adj-*Adjective*
3. prep-*Preposition*
4. adv-*Adverb*
5. conj-*Conjunction*
6. d-*Determiner*
7. pr-*Pronoun*
8. p-*Particle* (ligatures are included under this category)
9. v-*Verb*

Once the language and part of speech annotation was complete, the data was separated and retained separately:

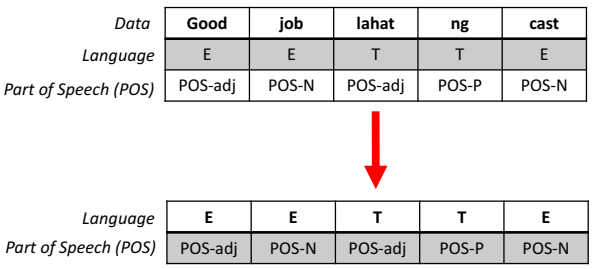


Figure 3.2 After annotation, the words themselves were held separately leaving just the language and the part of speech. Note that the original data is retained so we can map the parse back to it. Above: data with language and part of speech Below: language and part of speech (input to the parsing system)

3.2 Method

In this section, the process taken to transform the data to the parsed data is outlined. In figure 3.2.1, we see the parsed output is in the form of bracketed constituents. Before beginning, some information about constituents and underlying assumptions are mentioned below.

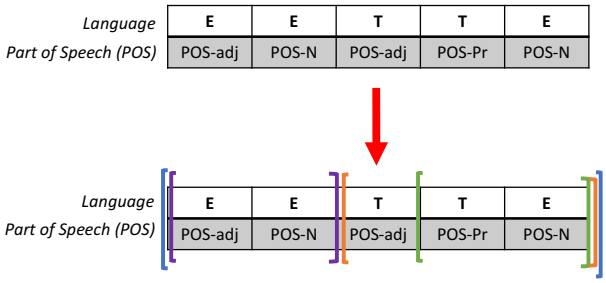


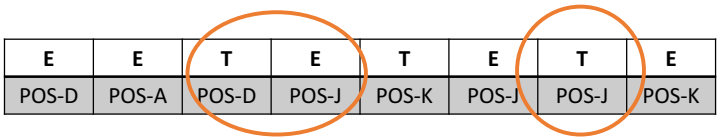
Figure 3.3 The output of the method will be the original input with brackets around constituents.

Underlying assumptions:

1. As mentioned in previous chapters, a constituent represents a group of words that act together as a single unit. A constituent in this thesis will be a sequential sub-sequence of the original natural language data sequence. This means it will be made up of words directly next to each other. For example, we could see a constituent like the following:

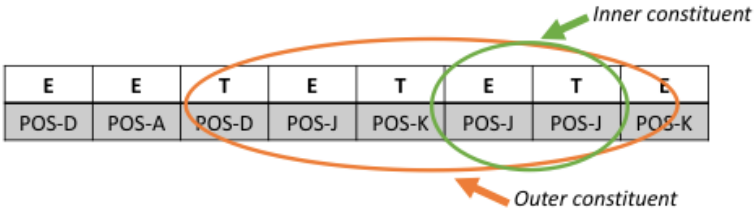
E	E	T	E	T	E	T	E
POS-D	POS-A	POS-D	POS-J	POS-K	POS-J	POS-J	POS-K

where the constituent is a direct, sequential sub-sequence of the starting data. We will not, however, see a "split up" constituent (meaning not a sequential subsequence):



This would not happen-should be directly sequential

2. A constituent can be nested within another constituent [cite]**.** This means that an outer constituent can contain another constituent within:



Additionally, this nesting can happen more than once, as shown below:

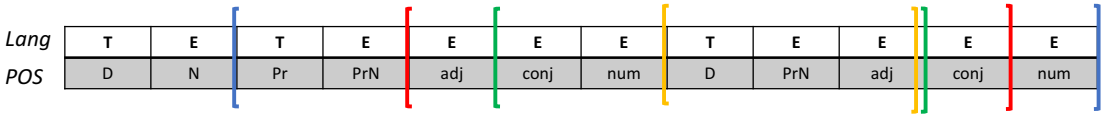


Figure 3.4 In this example, we can see the outer brackets (in blue) with nested constituents within (represented by red, green and yellow brackets). Note that the inner nested constituents can nest additional constituents and so on.

Before we can begin grouping words together into constituents, we must first define what creates a constituent. Instead of a using a grammar, we are using the data alone to group words together. In this data-driven method, we will look for relationships between word features to decide if they belong together. Specifically, we will look at the adjacency frequency of a word by part of speech by language. Once we have these frequencies, we can parse the natural language data by grouping together words that have high frequencies of occurrence.

3.2.1 Finding Relationships Between Words

The first step was to prepare the input and transform the annotated data (actual word, language, part of speech) to just the language and part of speech (the actual data is retained for later mapping the parse back to the data).

Data	**Good**	**job**	**lahat**	**ng**	**cast**
Language	E	E	T	T	E
Part of Speech (POS)	POS-adj	POS-N	POS-adj	POS-P	POS-N

Language	**E**	**E**	**T**	**T**	**E**
Part of Speech (POS)	POS-adj	POS-N	POS-adj	POS-P	POS-N

After this step was completed for all tweets, the next step was to randomly split the 11,981 tweets into two groups. The first group, the training set, was made up of 7,188 tweets and the second group, the test set, was made up of 4,693 tweets.

Each tweet in the training set was used to calculate the frequency of 36 unique bigrams. A bigram is a sequence of two adjacent tokens:

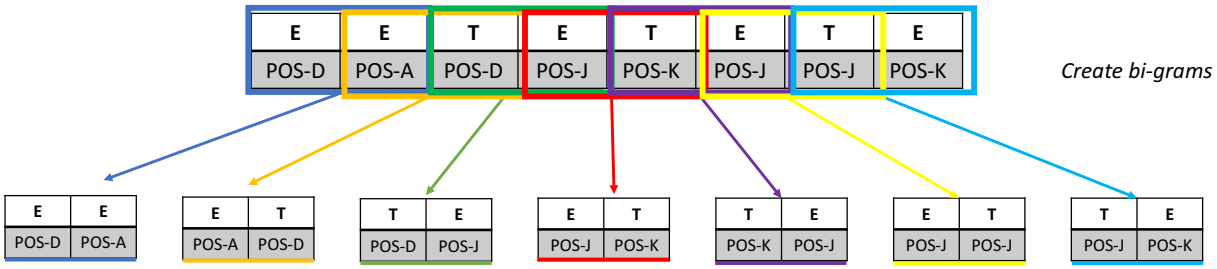
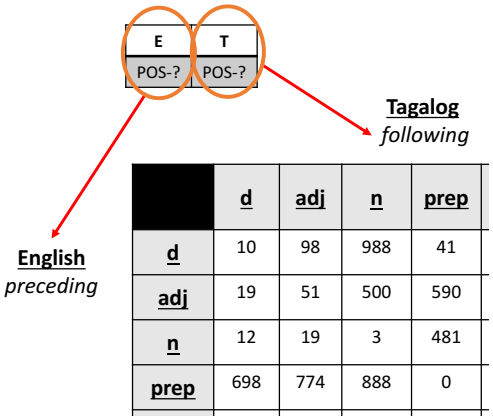


Figure 3.5 The original string of tokens used (top) to create bigrams (below). Note that a tweet composed of multiple sentences would have a bigram per sentence.

There were 36 unique bigrams created. Once the bigrams were created, likeliness adjacency for each possibility (language and part of speech) was calculated using the language and part of speech. For example:



	d	adj	n	prep
d	10	98	988	41
adj	19	51	500	590
n	12	19	3	481
prep	698	774	888	0

Figure 3.6 Part of a frequency chart representing the relationships between two words by language and part of speech. Relationships for all the following bigrams were found: English, Tagalog/ Taglog, English/ Tagalog, Tagalog/English, English

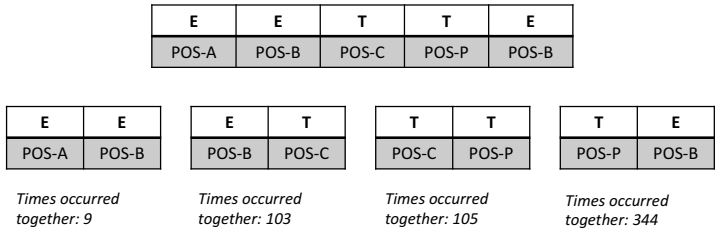
In figure 3.6, we see the adjacency frequency of a specific part of speech in English followed by a Tagalog word of a specific part of speech. With these values, discoverable relationships between words in a sequence are evident[8]. After examining all the data with the actual words, three human judges determined that bigrams with less 29 occurrences indicated that sequence was not acceptable. This determination was made by looking at bigrams (with the actual word data included) and discovering that all cases judged unacceptable by a human judge were less than a certain value (29). For example, an English noun followed by a Tagalog noun only occurred three times in the whole data set. Since this number was below 29, it would not be considered an acceptable grouping when building a constituent.

3.2.2 Grouping Constituents

Once the relationships between words based on language and specific parts of speech have been determined, we start building constituents out of our test data. This phase is composed of two main steps: the initial step (building the lowest level constituents) followed by continuously grouping the constituents

[8] Possible issues with this technique did not go unnoticed. For example, in *I see the dog the cat and the bird* (a possible natural language string that did not include the comma between *the dog* and *the cat*, meaning a determiner followed a noun), we see a determiner *(the)* followed by a noun *(dog)* followed by a determiner *(the)*. According to English grammar, *the* is not allowed to follow a noun, but if this occurred many times the frequency count would increase (and falsely lead us to group together a noun followed by a determiner). Another issue is that the frequencies are fitted to the data; like most data driven techniques, more data allows for more insights.

until we end up with a final, single constituent. We start at the beginning of our string and create bigrams. Note that in the following example, I am using toy parts of speech (A, B, C, P).



From these bigrams, we can start comparing neighboring words to see whether or not they group together to build constituents. We check if the first two words belong together (defined above 29 occurrences in the previous section):

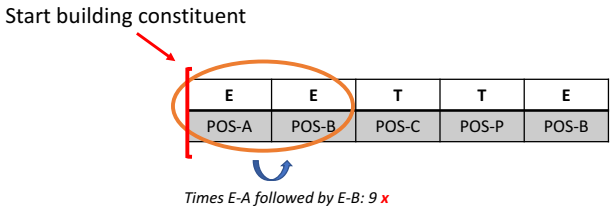


Figure 3.7 The first word starts off the first constituent. Since the frequency of the bigram is below 29, the first and second word do not group together as a constituent.

The first two words do not group together, so we end the constituent:

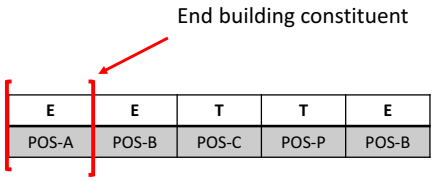
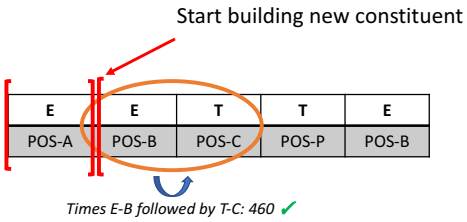
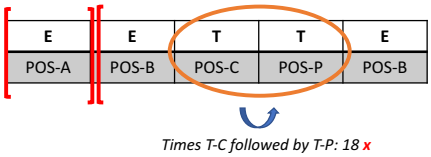


Figure 3.8 The constituent ending is indicated by a bracket

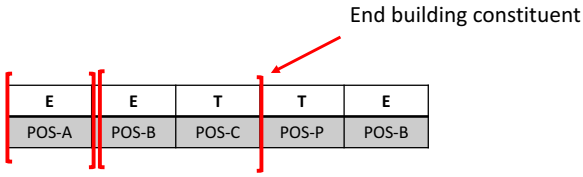
We now start building the next constituent and check the following bigram:



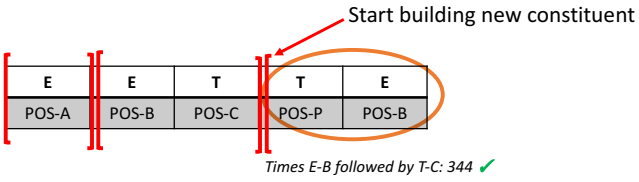
This combination occurred more than 29 times, so we group the two words together. We continue to the next bigram:



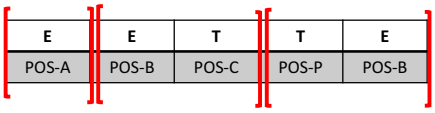
The frequency value is less than 29, so the two words do not group together. We are done with this constituent:



We continue and compare the last bigram. The frequency is greater than 29, so the last two words group together:



We are now done, so we close the final constituent:



The first step is complete since we have no more words to group. The next step is to continuously combine all constituents until we have one single constituent. We do this by using the same frequency table from the previous section between boundaries of the constituents:

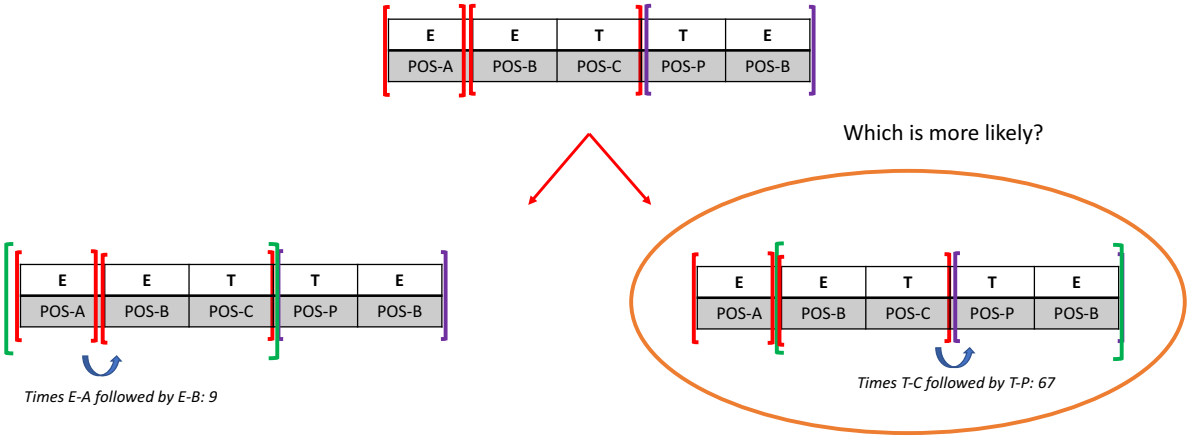
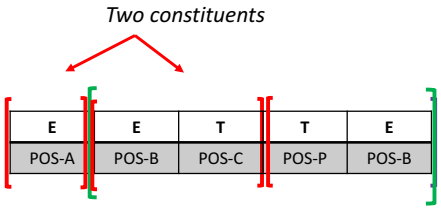
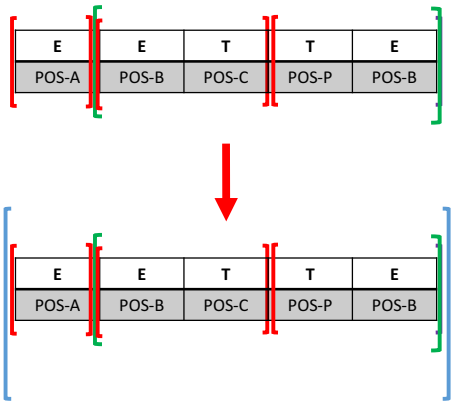


Figure 3.9 We now make a decision between either grouping the first and second constituent or the second and third constituent together

In this case, the right-hand grouping has a higher frequency between constituent boundaries, so the second and third constituent are grouped together in one constituent. We now have two constituents:



The final combination is putting the two constituents together:



We now map back this parse to the original language data.

3.3 Evaluation and Results

In constituency parsing, one commonly used method of evaluation is to start with a gold standard and give partial credit for correct parsing. A gold standard would be the acceptable parse for a given sentence.

The evaluation method of accuracy was as follows:

1. Each test tweet was divided into sentences. Each sentence was parsed by three human judges to create a gold standard. Total number of constituents were counted.

2. After the parse, the total number of constituents correctly identified for each sentence was counted. Each sentence received a score that was calculated by dividing the correct number of created constituents by the total number of constituents.

   $Score_{sentence}$= number of correctly created constituents/ total number of constituents

In terms of scoring, the human judges decided that a passing score was 80% or higher or more. These were the sentences that they felt were the most acceptable as they felt the most natural. Out of all the parsed sentences, we had 41.7 % with a passing score.

4. Conclusion

Below, I discuss the results from the previous section. I then look at possible future directions for research.

4.1 Discussion

While the data-driven method used in this thesis yielded some success, there is still room for substantial improvement. Firstly, the annotation step can and should be automated. This means that instead of having human annotators for language and part of speech identification, a computational technique would complete this task. A possible method for this step would be to use the monolingual resources available for both English and Tagalog. These resources, despite the low-resource status of Tagalog, would offer far more assistance than any currently existing Taglish resources. Additionally, in terms of a human cost component, this would save substantial time and effort while delivering a more comprehensive computational process.

The next step for improvement would be to move towards a solution with a machine learning component in the constituent building step. This would allow for a more robust overall system. Finally, the evaluation process could be improved by depending less on human judges. While human judgement is a useful tool, a stronger solution might require less human participation than used in this process. Ideally, future work on this topic should have a much larger computational element.

4.2 Future Directions

Working on this topic brought to light different aspects of processing code-switched natural language data and raised a few potential topics to be pursued in the future.

Firstly, with regards to the actual data used in processing code-switched natural language, a future endeavor involving parsed Taglish should possibly include work with intra-word switches. Intra-word

switches occur within the actual boundaries of a word, such as if a word is composed of different morphemes. For example, Taglish words like *nakakamiss* and *magdradrive* are examples of intra-word switching because English morphemes and Tagalog morphemes are combined to make one word. This means that a code-switch is between the morphemes and happening within the word itself. Specifically, both of these examples show a Tagalog prefix (*nakaka-* and *mag-*) attached to an English verb. Note that the English verb in the second example is preceded by a repeat of the first verb syllable (*dra* precedes *drive*). This is a Tagalog conjugation rule when dealing with verbs in the present and future tense.

Considering code-switched parsing starting at the morpheme level and above, instead of just the word level and above, could allow us to view the word itself as composed of constituents (morpheme constituents combined in a larger word constituent). We can then eventually pull out meaning embedded per morpheme, not per word. For example, the meaning of *nakaka-* does not have a direct translation in English. By breaking up the word into two parts, the Tagalog prefix *nakaka-* and the English verb *miss*, we can derive meaning for the whole word by understanding the meaning of *nakaka-* and *miss* in their respective languages and then looking for the final meaning in how they are combined. The prefix *nakaka-* modifies the verb it is attached to and thus modifies the English verb to give a different overall meaning than *miss* in English alone gives. Since one of our first steps in understanding natural language is to look for the most accurate meaning encoded in groups of words, deriving a composite final meaning from the different languages used within a word can help provide a more accurate meaning.

Another interesting consideration could be parsing when the code-switching includes emojis. As mentioned in chapter three, smileys, emojis and emoticons were removed from the natural language data for our purposes. Since code-switched data tends to happen in more relaxed and friendly environments, emojis, at this period in time, are a frequent addition into more casual natural language. Considering that a substantial amount of the data seen on Twitter during the research for this thesis contained emojis, it is not unfathomable to use emojis as part of the parsing process in the future. For example, the following examples were seen on Twitter:

1. 😀 na 😀 ako.
2. 😀 ako.

where the first example uses the Taglog structure to express "very" by using two emojis connected by *na*. Assuming the use of a wide-smiled emoji to indicate happy, the first example could translate to *I'm very happy* while the second example could translate to *I'm happy*. It might even be possible to consider the use of emojis as a language itself, meaning the code-switch would include switching between this emoji language and the natural language text.

As for learning and deriving information from code-switched natural language data, an interesting future study might be to look for a grammar from the data. Rather than just use the data alone to decide how to parse the data, a practical first step might be to try to derive the actual underlying grammar from the data. Once we can figure out that actual grammar, we can use it to help parse the language successfully. Note that finding a grammar and using it to parse alone might not be satisfactory. As stated in the introduction, natural language data does not effortlessly lend itself to being parsed from a grammar alone due to frequent ungrammatical use. A possible subsequent step after deriving an initial grammar might then be to use it as a component in a parsing system along with data-driven parsing.

Finally, another possible step after an initial grammar discovery would be to have a system that can continuously refine the grammar. With the addition of new data and exposure to new underlying grammar structures, this grammar would be able to adjust itself. Essentially, this would mean working with a grammar that is not rigid and can learn over time. This flexibility could possibly handle the unstructured nature of natural language, particularly code-switched data.

References

[1] C. D. Morrison, "Code-switching," *Encyclopædia Britannica*, 30-May-2017. Accessed on: 05-Aug-2019. [Online]. Available: https://www.britannica.com/topic/code-switching.

[2] D. Grune and C.J.H. Jacobs, *Parsing Techniques: A Practical Guide.* New York: Springer Science+Business Media LLC, 2008.

[3] J. Nivre, *Inductive Dependency Parsing*. The Netherlands: Springer, 2006.

[4] T. Solorio and Y. Liu, "Learning to Predict Code-Switching Points," In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pp. 973-981, 2008.

[5] M.L.S Bautista, "Tagalog-English Code Switching as a Mode of Discourse," *Asia Pacific Education Review*, vol. 5, no. 2, pp. 226-233, 2004.

[6] C. Myers-Scotton, *Dueling Languages: grammatical structure in codeswitching*. New York: Oxford University Press, 1993.

[7] D. Sankoff and S. Poplack, "A Formal Grammar for Code-switching," *Papers in Linguistics: International Journal of Human Communication*, vol. 14, no. 1, pp. 3-46, 1981.

[8] R. Wardhaugh, *An Introduction to Sociolinguistics,* 5th ed. Oxford:Blackwell Publishing, 2006.

[9] S. Parveen and S. Aslam, "A Study on Reasons for Code-Switching in Facebook by Pakistani Urdu English Bilinguals," *Language in India*, vol. 13, issue 11, pp. 564-590, 2013.

[10] S. Gal, "Language and Political Economy," *Annual Review of Anthropology*, vol. 18, pp. 345-367,1989.

[11] "Philippines," *Ethnologue*. Accessed on: 06-Jul-2019. [Online]. Available: https://www.ethnologue.com/country/PH.

[12] G.C. Borlaza, M. Cullinane, and C.G. Hernandez, "Philippines," *Encyclopædia Britannica*, 17-Aug-2018. Accessed on: 05-Aug-2019. [Online]. Available: https://www.britannica.com/place/Philippines.

[13] J.M. Lipski, "Remixing a mixed language: The Emergence of a New Pronominal System in Chabacano (Philippine Creole Spanish)," *International Journal of Bilingualism*, vol. 17, no. 4, pp. 448-478, 2012.

[14] A. Gonzalez, "The Language Planning Situation in the Philippines," Accessed on: 06-Sept-2019. [Online]. Available: https://web.archive.org/web/20070616101625/http://www.multilingual-matters.net/jmmd/019/0487/jmmd0190487.pdf

[15] V.L. Rafael, "Taglish, or the Phantom Power of the Lingua Franca," *Public Culture*, vol. 8, issue 1, pp. 101-126, 1995.

[16] F.M. Azores, "A preliminary investigation of the phenomenon of language change in the Philippines," Unpublished master's thesis, Ateno de Manila University, 1967.

[17] E. Manguilimotan and Y. Matsumoto, "Dependency-based Analysis for Tagalog Sentences," In 25th Pacific Asia Conference on Language, Information and Computation, pp. 343-352, 2011.

[18] R.E. Roxas et al., "Building Language Resources for a Multi-Engine English-Filipino Machine Translation System," *Language Resources and Evaluation*, vol. 42, no.2, pp. 183-195, 2008.

[19] R. Tsarfaty et al., "Parsing Morphologically Rich Languages: Introduction to the Special Issue," *Association for Computational Linguistics*, vol. 39, no. 1, 2013.

[20] J. Nivre et al., "MaltParser: A Language-Independent system for data-driven dependency parsing," *Natural Language Engineering*, vol. 13, no. 2, pp. 95-135, 2007.

[21] "The Stanford NLP Group," *The Stanford Natural Language Processing Group*. Accessed on: 06-Sept-2019. [Online]. Available: https://nlp.stanford.edu/software/nndep.html.

[22] D. Jurafsky and J.H. Martin, *Speech and Language Processing, 3rd edition draft.* [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[23] Ö. Çetinoglu, S. Schulz, and N.T Vu, "Challenges of Computational Processing of Code-Switching," In Proceedings of the Second Workshop on Computational Approaches to Code Switching, pp. 1-11, 2016.

[24] A. Taylor, M. Marcus, and B. Santorini, "The Penn Treebank: An Overview," Accessed on: 06-Sept-2019. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.8216&rep=rep1&type=pdf.

[25] M. Mistica and T. Baldwin, "Recognising the Predicate-argument Structure of Tagalog," In Proceedings of NAACL HLT 2009: Short Papers, pp.257-260.

[26] L. Kallmeyer and W. Maier, "Data-Driven Parsing using Probabilistic Linear Context-Free Rewriting Systems," *Association for Computation Linguistics*, vol. 39, no. 1, pp. 87, 2013

[27] H.B. Hashemi and R. Hwa, "Parse Tree Fragmentation of Ungrammatical Sentences," In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intellegence, 2796-2802.

[28] T. Solorio and Y. Liu, “Part-of-Speech Tagging for English-Spanish Code-Switched Text,” In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pp. 1051-1060, 2008.

[29] W.B. Cavnar and J.M. Trenkle, “N-Gram-Based Text Categorization,” In Proceedings of SDAIR-94,1994.

[30] M. Zampieri, N. Ljubešić, J. Tiedemann, and L. Tan, “Merging Comparable Data Sources for the Discrimination of Similar Languages: The DSL Corpus Collection,” In Proceedings of the 7th Workshop on Building and Using Comparable Corpora (BUCC), pp. 6-10, 2014.

[31] A.K. Joshi, “Processing of sentences with intra-sentential code-switching,” In Proceedings of COLING, pp. 145-150, 1982.

[32] A. Sharma et al., “Shallow Parsing Pipeline-Hindi-English Code-Mixed Social Media Text,” In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computation, pp. 1340-1345, 2016.

[33] D. Vilares, C. Gomez-Rodriguez, and M.A. Alonso, “One model, two languages: training bilingual parsers with harmonized treebanks,” In Proceedings of 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 425-431, 2016.