– Independent Work Report Fall 2022 –

Constructing an Automated Lemmatizer and Part-Of-Speech Tagger for Asante Twi

Jeremy Dapaah
Adviser: Dr. Christiane Fellbaum

January 12, 2023

Abstract

We build a deterministic lemmatizer for verbs in the Asante Twi dialect of the Akan language. Using the Google Cloud Translate API to confirm the existence of possible words, and part-of-speech taggers from the Natural Language Toolkit and the Stanford NLP Group to tag the words in a unigram model, we establish and compare rudimentary part-of-speech taggers for Asante Twi with a maximum accuracy of 67%.

1. Introduction

As the world becomes increasing more globalized, the need to quickly translate ideas between languages becomes paramount, and such tasks often need to be accomplished faster than the speed of a human translator. Machine Translation (MT) is a sub-field of computational linguistics that encompasses the translation of text and speech between different human languages by computers. The potential application of computers for language translation dates back to the advent of modern computers in the 1950s [25]. Since then, the leading edge technologies in MT have evolved multiple times, from rule-based methods to statistical models. Recent advancements in neural MT, which uses machine learning models such as recurrent neural networks, have been accomplished by multiple companies including Google and Microsoft. The implementation of such systems is often accomplished via the use of training via tremendous amounts of data in large language models (LLMs). Translators built with LLMs have demonstrated an impressive degree of accuracy across tens of languages, but often struggle with small details within a sentence that can often be important. Additionally, many of the nuances and inflections in a language are deterministic, and do not necessarily need a machine learning model to train on their distinctions endlessly. This issue brings into focus the strategy of **hybrid MT**, which uses multiple established methods for machine translation in conjunction with each other. For example, rule-based MT techniques can be used to supplement and correct the output produced by neural MT methods such as Google's Neural Machine Translation (GNMT).

Outside of full translators, there exist other MT tools for natural language tasks such as document summarizing and sentiment analysis. One such example is a **lemmatizer**, which simplifies a word into the form which you find in a dictionary by removing prefixes and suffixes. A lemmatizer is a necessary component in constructing a WordNet. A WordNet is a large lexical database that organizes English words and their synonyms in a tree structure [11]. By training on word embeddings, the tree is constructed such that words that occur in similar context are close in proximity occur to each other. Such a tool can be used to discern the difference in homonyms, which leads to a better understanding of the context of a text passage. WordNets are created for individual languages; the first was created for the English language, other have been created, such for Russian [23]. Future work involving the construction of a Twi WordNet would be advantageous, as Twi possesses many homonyms that serve different functions in the sentence.

Another useful MT tool is a **part-of-speech tagger**. A part of speech (POS) tagger annotates the tokens in a sentence with their corresponding part of speech. Such tools are useful because they generate explicit metadata about the components of a sentence. All languages have syntax: rules that are followed in order to properly structure and format an idea. While syntax is not a immovably rigid framework, understanding it and how it applies to a sentence is critical. If a natural language processing (NLP) pipeline sends the output of a POS tagger to a natural language parser, a MT tool that identifies the role each token plays in a sentence [6] such as the subject or indirect object, knowing the object of the previous sentence can provide contextual metadata for the next sentence in determining what is referenced by a personal pronoun.

Such tools are not commonly available for African languages, which are recently more frequently coming into the fold of NLP advancements. The goal of this project is to construct a lemmatizer and a POS tagger for the Asante Twi dialect of Akan, a language spoken most commonly in Ghana, West Africa. In summary, the lemmatizer is used to generate potential roots in Asante Twi, which are filtered and tagged via their English translations. This work then compares the results of two separate POS tagging systems, those of the Stanford NLP Group and the Natural Language Toolkit (NLTK). The results of the system were then compared against human POS tagging, and had accuracies of 67% and 64%, respectively.

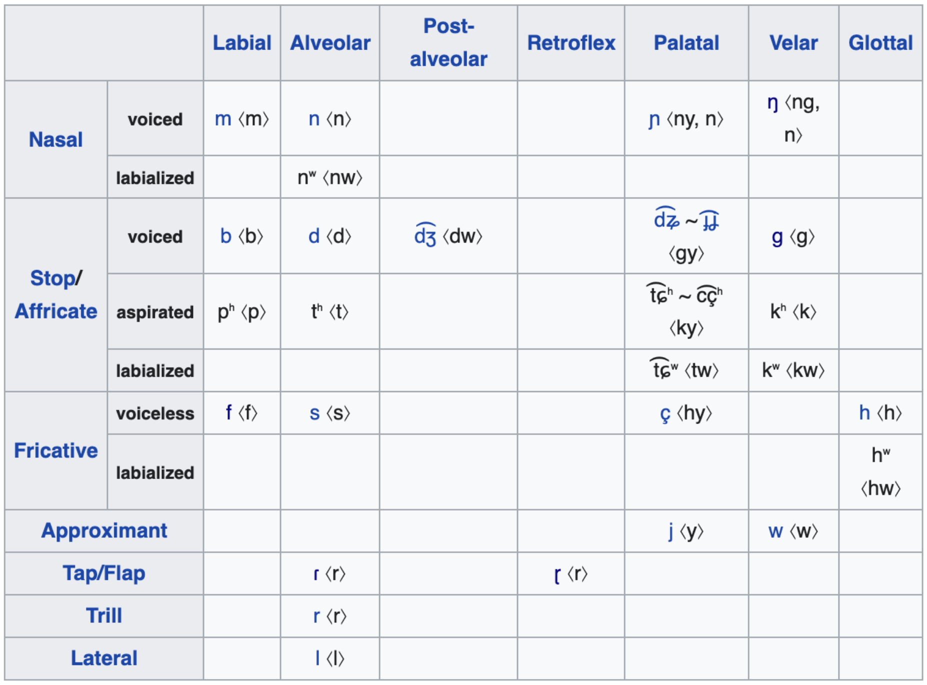
2. Background

2.1. Akan & Asante Twi

Akan is a Niger-Congo language spoken in Ghana and the surrounding countries of Cote D'Ivoire and Togo by approximately 20 million people [1]. While English is the official language of Ghana, the dialects of Akan are still heavily used in spoken conversation, and are common for most Ghanaians to speak on a regular basis. The Asante Twi dialect of Akan is spoken primarily in Ghana, and as one of the three major dialects of Akan, the language is mutually intelligible with Fante and Akuapem Twi. Relative to other dialects, the Asante dialect is spoken most commonly in the southern region of Ghana, is the overall most common dialect of Twi [27].

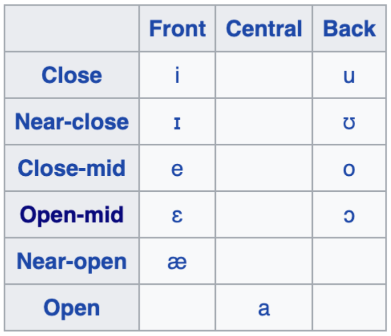
2.1.1. Phonology and Phonetics Asante Twi uses the consonantal and vocalic phonemes illustrated in Figures 1 and 2, respectively. Figure 1 includes in angle brackets how each phoneme is typically written in the orthography. For the vowels, the typical orthographic representation of each of the phonemes is shown in Table 1. There, we see the influence the introduction of the Latin alphabet had on Twi's written language; many of the letters are borrowed the the International Phonetic Alphabet (IPA).

Like most West African languages, Twi employs the use of tones in its vowels. Differences in tone can change the meaning of words, such as in the difference among the words in Table 2. While the three words are written with the same letters and possess the same speech sounds, the difference in tone dictates a different meaning to each word.



		Labial	Alveolar	Post-alveolar	Retroflex	Palatal	Velar	Glottal
Nasal	voiced	m ⟨m⟩	n ⟨n⟩			ɲ ⟨ny, n⟩	ŋ ⟨ng, n⟩	
	labialized		n^w ⟨nw⟩					
Stop/ Affricate	voiced	b ⟨b⟩	d ⟨d⟩	d͡ʒ ⟨dw⟩		d͡ʑ ~ ɟ͡ʝ ⟨gy⟩	g ⟨g⟩	
	aspirated	p^h ⟨p⟩	t^h ⟨t⟩			t͡ɕ^h ~ c͡ç^h ⟨ky⟩	k^h ⟨k⟩	
	labialized					t͡ɕ^w ⟨tw⟩	k^w ⟨kw⟩	
Fricative	voiceless	f ⟨f⟩	s ⟨s⟩			ç ⟨hy⟩		h ⟨h⟩
	labialized							h^w ⟨hw⟩
Approximant						j ⟨y⟩	w ⟨w⟩	
Tap/Flap			ɾ ⟨r⟩		ɽ ⟨r⟩			
Trill			r ⟨r⟩					
Lateral			l ⟨l⟩					

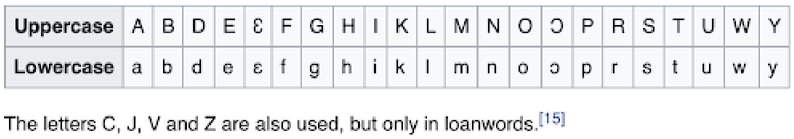
Figure 1: Phonemic consonants used in Asante Twi [32].



	Front	Central	Back
Close	i		u
Near-close	ɪ		ʊ
Close-mid	e		o
Open-mid	ɛ		ɔ
Near-open	æ		
Open		a	

Figure 2: Phonemic vowels used in Asante Twi [32].

2.1.2. Orthography Asante Twi is primarily spoken, rather than written, but uses an orthography with predominantly Latin characters.



Uppercase	A	B	D	E	Ɛ	F	G	H	I	K	L	M	N	O	Ɔ	P	R	S	T	U	W	Y
Lowercase	a	b	d	e	ɛ	f	g	h	i	k	l	m	n	o	ɔ	p	r	s	t	u	w	y

The letters C, J, V and Z are also used, but only in loanwords.[15]

Figure 3: Orthographic symbols used in Asante Twi [32].

Twi orthography is not taught or used extensively past grade school; students usually learn the orthography and remember it but do not use it frequently in common life, as most written education past that point is conducted in English.

IPA phoneme	Orthographic Representation
i	i
ɪ	i
e	e
ɛ	ɛ
æ	a
a	a
ɔ	ɔ
o	o
ʊ	u
u	u

Table 1: Mapping of Vowel Phonemes to Orthography

Twi Word	Tone Pattern	English Gloss
pápá	High - High	"good"
pàpá	Low - High	"father"
pàpà	Low - Low	"fan"

Table 2: Example of tonal minimal pairs in Asante Twi [7]

2.1.3. Parts of Speech Previous analyses of the Twi language identify the following 8 parts of speech: nouns, verbs, adjectives, adverbs, pronouns, numerals, conjunctions, and interjections [17]. However, for the purposes of this project, we include two additional parts of speech: Prepositions and Determiners. In Twi, prepositions and determiners are often represented using words such as *na*, and *no*, which have multiple other use cases. Including them allows the results to better understand the weaknesses of the model.

2.1.4. Morphology Morphology is the linguistic study of how words change, when new components (called morphemes) are connected. The process of adding morphemes can change characteristics including the number of a noun, the tense of a verb, or even a word's part of speech. An English example of a morphological change is the transition of a verb to a noun by adding the suffix morpheme "-er". This example is seen in the verb "climb" to the noun "climber"

Twi's morphology is most abundantly seen in the expression of its verb tenses. The written expression of each of the tenses is included in Table 3. In multiple tenses, the /n/ phoneme is added as a prefix to the root verb. This addition can trigger multiple phonological changes that modify the root or the new /n/ phoneme itself to make the word easier to pronounce.

$$/n/ \rightarrow [+nasal]_{\alpha-PoA}/\#\_C_{\alpha-PoA}$$

This phonological rule will change the /n/ in the prefix so that it matches the place of articulation of the first sound in the root word. For example, the /n/ prefix will be realized as an [m] when followed by a /b/, /p/, /m/, or /f/, all of which are articulated at the lips. When the root word starts with a /k/ or /g/, the /n/ is realized as an [ŋ], which is articulated at the velum. However, as seen in Figure 1, [ŋ] remains written as ⟨n⟩ in the writing system.

$$\begin{bmatrix} +voiced \\ -sonorant \\ -continuant \\ -dorsal \end{bmatrix} \rightarrow [+nasal]_{\alpha-PoA}/\#[+nasal]\_{}_{\alpha-PoA}$$

This rule changes the beginning of the root to match the place of articulation of the nasal sound if the root begins with a /b/ or a /d/. For example, adding the prefix /an/ to the verbs *ba* or *da* result in the new verbs *amma* and *anna*, respectively. An extended look at examples of the morphological changes for each of the verb tenses is included in the Appendix in Table 5.

Nouns, in general, use the same morphological change as the present negative and future negative verb tenses to denote pluralization (e.g., the plural form of the noun *ba* (“child”) is *mma* (“children”).

2.2. Lemmatization

Lemmatization is the process by which words are turned into their base form by stripping them of their inflectional morphemes (e.g. turning “playing” into “play” or “best” into “good”). Perfect lemmatization requires knowledge of the part of speech that a word takes in a sentence, as well as the definition of the word and its lemma. For example, knowing that the word “miss” is a verb prevents an incorrect lemmatization of removing the final “-s” to turn a plural noun in the singular noun “mis”. Lemmatization in Twi is a delicate process, as the meanings of words in the language

Composition	**Tense**
{verb}	Present
/bɛ/ + {verb}	Future
/re/ + {verb}	Progressive
/rebɛ/ + {verb}	Immediate Future
{verb} + {X}	Past with complement
{verb} + /eɛ/	Past without complement
/a/ + {verb}	Present Perfect
/n/ + {verb}	Present Negative
	Future Negative
/ren/ + {verb}	Progressive Negative
	Immediate Future Negative
/an/ + {verb}	Past Negative
/n/ + {verb} + {X}	Present Perfect Negative with complement
/n/ + {verb} + /eɛ/	Present Perfect Negative without complement

Table 3: Morphological changes to denote verb tense in Asante Twi, where {X} represents a repeat of the last letter in the root

are heavily dependent on context. For example, attempting to lemmatize the Twi verb *mma* could result in two legal possibilities: the verb *ba* ("come") and the verb *ma* ("give"). This means that it would be impossible to generate a fully correct lemmatization of a text corpora without a method for determining the context of the full passage.

2.3. POS Tagging

A POS tagger annotates the tokens in a sentence with their corresponding part of speech. For example, in the sentence, "I ring the doorbell", a POS tagger would identify the pronoun "I", the verb "ring", the determiner "the", and finally, the noun "doorbell". POS tagging must also be accomplished with knowledge of the context in which a word appears. It may be ineffective to attempt to tag with a unigram model, as doing so would be unable to differentiate homonyms - the tagger would have no way to identify the difference between the noun "ring", a circular band worn around the finger, or the verb "ring", to make a continuous high-pitched sound. This problem is also relevant in Asante Twi - the word *ba* functions as the verb "to come", but it also exists in the language as an the noun "child".

There is an inherent recursive relationship between lemmatization and part of speech tagging.

Part of speech tagging often occurs by attempted lemmatization: if an English word ends in the suffix "-ing", it is likely to be a verb being represented in the progressive tense. However, the method of lemmatization is dependent on the part of speech - different rules are used to lemmatize verbs than those that are used to lemmatize adjectives, or nouns.

As is the case with most language translation pairs, Twi does not match to English on a word-by-word basis, so tagging efforts are rarely straightforward. In the Twi sentence *me kɔɔ sukuu* ("I went to school"), the English word "to" is added in the translation process in order to uphold English syntactic constraints. English taggers typically include a large amount of nuance in their tagging, using the IBM set of tags [3]. Twi's parts of speech are more general and at a higher level; most of the detail seen in English is abstracted away when we tag words in Twi.

3. Previous Work

3.1. Parallel Corpora

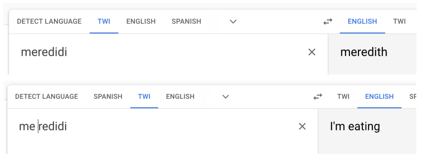
Much of the existing work in natural language processing for dialects of Twi is involved in generating parallel corpora for the language. Parallel corpora are textual datasets that translate sentences and/or paragraphs between two languages. These are useful because they provide labeled pairs for training translation models between two languages. Often, these works will also involve an application of the parallel dataset, such as the development of a translation technique or an evaluation of the dataset.

For example, work by Azurne et al. collected ∼25,000 sentence pairs in English and used a NMT system known as OPUS-MT to generate rudimentary translation from English to Akuapem Twi. Then, the generated sentences were verified or corrected by native speakers [14]. Other work by Adjeisah et al. has constructed a massively parallel corpus, which establishes pairwise parallel corpora between three or more languages. This corpus compares different versions of the Twi Bible with versions in English, Chinese, Spanish, Greek, and other languages [13].

3.2. Available Twi Translators

Another example of the application of neural MT to African languages is the introduction of Twi to **Google Translate**, which occurred recently in mid-2022 [16]. Early versions of Translate used phrase-based MT methods to translate languages, which would translate the sentence in fragments of words and phrases, and often result in *translationese*, a symptom of machine translation where the output sounds robotic and unnatural [24, 26, 14]. In the past, Translate would also use translate between two non-English languages in two steps, using English as a intermediary [15]. However, in 2016 Google updated Translate to perform with a novel NMT architecture known as GNMT, which is now executing on Translate's production environment for over a hundred languages[29, 24]. Translate uses a long-short-term-memory (LSTM) architecture, a form of recurrent neural networks [33]. Now no longer using the phrase-based architecture, entire sentences are used as input into the model, rather than word and phrases that can result in broken meaning [31]. Additionally, Google added "Zero-Shot" translation mechanisms to Translate, which use transfer learning to prevent the two-step translation problem and translate between the source and target language directly [29, 22].

Twi writing often combines the subject and verb when the subject is a personal pronoun. Google Translate, as an recent technology with Twi, does not perfectly interpret the lack of difference between split versus conjoined pronoun-verb constructions. Sometimes it only correctly identifies one (Figure 4), sometimes it correctly identifies both (Figure 5).



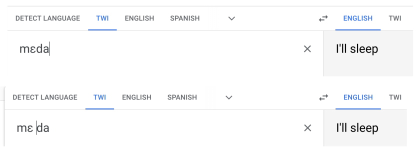
meredidi

meredith

me redidi

I'm eating

Figure 4: Only the split version correctly translates.



mɛda

I'll sleep

mɛ da

I'll sleep

Figure 5: Both versions are translated correctly.

It should also be noted that Google Translate does not yet have speech input or output designed for Twi. However, there are other existing translators which posses the ability to process speech input. Khaya is a translation tool built by Ghana NLP, which can translate between Akuapem Twi and English. It also possesses the ability to use one's speech in Twi as input to translate into English [20]. Ghana NLP is an open source initiative focused on natural Language Processing of Ghanaian languages [19]. Aside from Khaya, Ghana NLP have released other projects such as ABENA and BAKO, variants of BERT trained on Akan data. BERT, or Bidirectional Encoder Representations from Transformers, is a language representation model created by Google in 2018, which trains a general intelligence framework on unlabeled text. After training, an output layer can be added and fine-tuned to solve multiple NLP problems such as question-answering or language inference, without retraining the entire model for the specific task [18].

3.3. Available Machine Translation Tools

Stanford University has developed a suite of machine translation tools for multiple languages known as **Stanford CoreNLP (CNLP)**. This suite of tools includes a lemmatizer and POS tagger, a natural language parser, and a named entity recognition tagger, which can identify named entities such as people and locations. In addition to English, the CoreNLP project also has datasets and models for languages including such as Chinese, French, Arabic, and German.

The NLTK is an open source library of machine translation tools in Python [2]. In addition to both an English lemmatizer and a POS tagger, it also includes resources such as WordNet as well as a suite of libraries for tasks such as semantic reasoning, classification, and syntax tree creation. It also includes wrappers that can initialize a local server instance of the CNLP suite.

These works either focus on the high level goal of translation for Twi, or provide MT tools for non-Ghanaian languages - there are no publicly available MT tools for any dialects of Akan. This projects seeks to bridge that gap by developing lemmatization and POS tagging tools for Asante Twi. Additionally, in designing a POS tagger for Asante Twi, the statistics of the NLTK and CNLP taggers are compared.

4. Approach

Of the multiple dialects of Akan, I chose to work with Asante Twi, since it is spoken most commonly. Additionally, the native speakers of the language whom I was able to be in continual contact with, including relatives and course professors at Princeton University, are all speakers of Asante Twi. Lemmatization can occur for all parts of speech, but this project focuses on lemmatizing verbs, which have the widest range of inflectional and derivational morphology out of parts of speech in Asante Twi. This paper seeks to take a rules-based approach to constructing the lemmatizer. Lemmatization for verbs in Twi is extremely consistent, save for a few irregular verbs. Thus, it was deemed a better approach to make the lemmatization a deterministic process, rather than training a machine learning model on textual examples. The construction of a POS tagger for Twi is a novel objective, so this work aims at comparing two English POS taggers in a pipeline for tagging Twi words. We will be comparing the taggers from CNLP and the NLTK.

4.1. Text Corpus

The corpus is composed of text excerpts taken from classwork from the Twi language course sequence at Princeton University, which is taught and created by Professor Hannah Essien, a native speaker of Asante Twi. The corpus includes 1083 tokens and 449 types.

4.1.1. Preprocessing the Corpus The course text corpora required cleaning in order to be machine readable. The additional, non-English characters ɔ and ɛ are not universally consistent throughout the corpus, as it includes other very similar characters from Unicode [10, 8, 9]. The characters were standardized to be sanitized to Unicode #596 and Unicode #603 for the lowercase, and Unicode #390 and #400 for the uppercase variants. This correlates to the same Unicode symbols used by the IPA, which is where the Twi alphabet is sourced from. The preprocessing was accomplished by filtering out and replacing any instances of wayward characters, which were anything other than the orthography [Figure 3] and standard punctuation.

In order to evaluate the POS tagger, it was necessary to assign parts of speech to the tokens in the corpus. This was done by hand as we applied parts of speech to the Twi text based on the English translations of the text, which was provided by native speakers. The labeled data is included in this project's git repository as corpora/course-text_LABELED.txt.

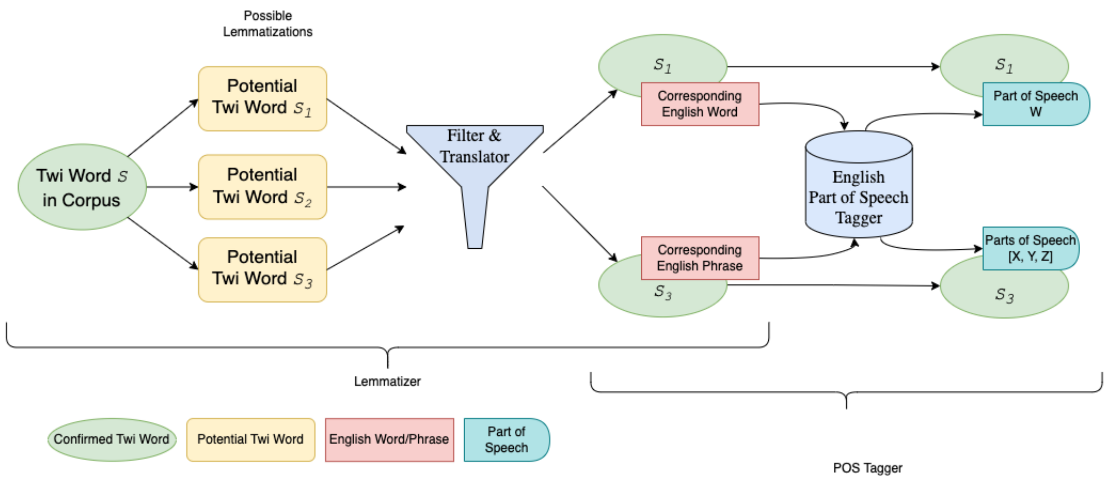


Figure 6: A general diagram of the pipeline for lemmatizing and tagging Twi words

4.2. Overview of the Pipeline

As shown in Figure 6, for each word in the corpora, the lemmatizer will attempt the twelve different forms of verb lemmatization. This approach is similar to the approach taken by the NLTK English lemmatizer, which holds a list of the most frequent morphological endings and removes them according to which element they match in the list. The NLTK's list of endings is included in the Appendix at Figure 9. The forms that are successfully lemmatized into root words will then be filtered to collect all the words that exist in the Twi dictionary.

The Twi POS tagger will use the English translation of the Twi word as input into the POS tagger. We can then attribute the parts of speech returned from the tagger to the original and lemmatized Twi word. Since Twi uses fewer parts of speech than the NLTK tagger, we will map the larger English set to the more general subset we will use for Twi, described in Section 2.1.3.

Since Akan verbs in the present tense are in their root form, the lemmatizer sends the unmodified word to the POS tagger as a potential present tense verb. The set of words that are labeled 'present tense verb' includes all words that are not verbs, since those words were not modified in the lemmatization process, so the lemmatizer's default value of 'present tense' will be discarded if the POS tagger returns a POS other than that of a verb.

5. Implementation

In order to collect the possible lemmatizations of each Twi word, all of the different lemmatization options were attempted. The possibilities that matched a tense's morphology were collected, and then were filtered to see if the lemmatized result translated to an English word. Originally, the lemmatization is considered successful only if the English word returned is different from the word in Twi. However, this causes an issue with idempotent **loanwords** from the English language. Loanwords are words that are introduced to a target language that has no existing corollary. As a result, the word is typically modified from the source language to fit the phonological and orthographic constraints of the target language. An example is the word *sukuu* in Twi, which translates to the word "school" in English. However, some loanwords from the English language are not re-transcribed in the orthography, such as more specific and proper nouns, such as "biology", or "Princeton". Given the original filtering process, such words would be removed form the set of lemmatized words as attempting to translate them from Twi to English is an idempotent process and has no effect on the word. This would cause many loanwords to erroneously never be included as a word used in Twi. To resolve this, the filtration process was then modified to append the original word to the list of successful lemmatizations of successful lemmatizations if no other options were available. At the very least, the word must exist in Twi as written.

As mentioned, implementing the filter requires a system that can translate from Twi to English. Currently, the best system available to do so is **Google Cloud Translate**. I elected to use a cloud service because it allowed me to access the translation via an API. Other options included building a web scraper module to read the data from a website, but this was deemed unnecessary and inefficient. All existing translation sites that included Akan were built for web browser consumption and were too overstuffed with HTML elements to warrant attempting to scrape. Additionally, most services that translate Akan are using Google's GNMT API in the background, as it is the only MT service widely available that includes Akan. As such, my decision to use a cloud service necessitated that I choose Google Cloud Translate. Using Google Cloud Translate required that I create a service account within a project, and enable the Cloud Translate API [4]. Once I created the service account, I created a private key stored in the git repository as ‘service-account.json‘. The client authenticates via the key in order to make requests to the Translate API.

5.1. Comparison of the Part of Speech Taggers

5.1.1. CNLP Tagger The Stanford CoreNLP tagger was originally constructed in 2003 by Kristina Toutanova and Chris Manning using a maximum entropy-based model, but has been improved upon in speed and usability since becoming publicly available. At its genesis, it performed at an accuracy of 96.9% on the WSJ dataset, and at 86.9% on unseen data [30].

To evaluate the CoreNLP tagger, our evaluation script initially would use the Java package available for public use [5]. However, since the project package in written in Python, using the Java package required calling a subprocess for every word in the corpus, which was very slow and expensive. Thus, we chose to use an NLTK wrapper script to communicate with a connection to a CNLP server instance running the Java package in the background [12]. This improved the speed of our evaluation more than twofold, as the evaluation script no longer needed to write files to the disk and spawn subprocesses. Additionally, the server instance responds to requests on multiple threads, as opposed to the single threaded structure of the Java package in a subprocess.

5.1.2. NLTK Tagger The NLTK POS tagger was built by Matthew Honnibal in 2013 using an Averaged Perceptron deep learning model [21]. It performs at an comparable accuracy of 96.8% on the WSJ dataset, and 91.8% accuracy on unseen data.

5.2. Evaluation Metric of the Part of Speech Taggers

In Section 2.3, it was discussed that Twi words and English words are not bijective - they do not match one-to-one. This fact becomes especially relevant when evaluating the part of speech taggers. Twi words will often translate to a phrase in the English language, the words of which each have their own individual part of speech. For our evaluation metric, the Twi tagging module is considered correct for word *S* if the true Twi tag, TRUE_POS(*S*), is included among any of the English tags returned by the tagger.

$$\text{TRUE\_POS}(S) \in [W,X,Y,Z] \implies \text{PRED\_POS}(S) := \text{TRUE\_POS}(S)$$

However, in evaluating incorrect categorizations, the Twi word must be categorized under a single part of speech. This requires a single part of speech to be chosen across any English phrases returned from the Translate API, as well as between other lemmatizations. The following ranking was decided by the approximated strength of a part of speech based on my experience constructing linguistic syntax trees: Verbs, Nouns, Pronouns, Adverbs, Adjectives, Numerals, Determiners, Conjunctions, Prepositions, and Interjections.

$$\text{TRUE\_POS}(S) \notin [W,X,Y,Z] \implies \text{PRED\_POS}(S) := \text{STRONGEST}([W,X,Y,Z])$$

6. Results

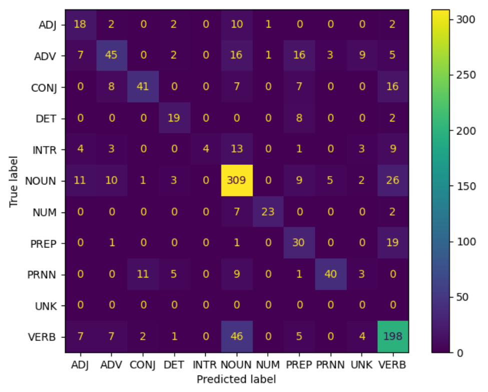


Figure 7: Confusion Matrix for the CoreNLP Tagger [28].

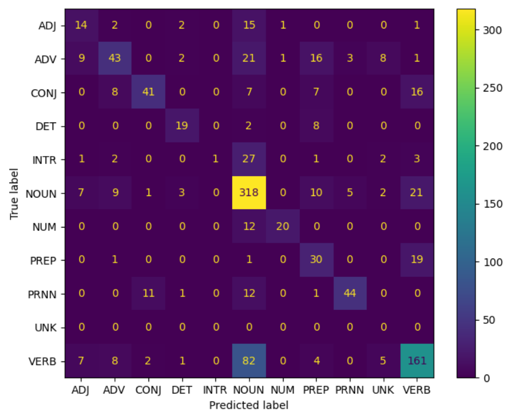


Figure 8: Confusion Matrix for the NLTK Tagger [28].

This results section compares the POS predictions from the Twi taggers and compares their results against the labels assigned by humans. The raw confusion matrices for each tagger are given

Part of Speech	CNLP Precision	NLTK Precision	CNLP Recall	NLTK Recall
ADJ	0.383	0.368	0.514	0.4
ADV	0.592	0.589	0.433	0.413
CONJ	0.745	0.745	0.519	0.519
DET	0.594	0.679	0.655	0.655
INTR	1.0	1.0	0.108	0.027
NOUN	0.739	0.64	0.822	0.846
NUM	0.92	0.909	0.719	0.625
PREP	0.39	0.39	0.588	0.588
PRNN	0.833	0.846	0.58	0.638
VERB	0.71	0.725	0.733	0.596

Table 4: Precision and Recall Statistics of the Taggers

in Figures 7 and 8. The precision and recall measurements for each tagger are listed in Table 4, with the higher value for each comparative statistic highlighted.

Overall, the both taggers were reasonably successful with pronouns, nouns, numerals, and verbs, both with precision values above 50%. This makes sense in the case of nouns. Nouns are most likely to be unique to Twi and not be changed when attempted to be translated to English. Additionally, POS taggers will often default to nouns when encountering rare words, leading to nouns correctly represented when working with the new language of Twi.

However, both taggers were exceptionally poor at identifying interjections. Both taggers had a precision score of 1, but this is only the case because they correctly identified four or fewer interjections, out of the 37 that were in the dataset. Evaluating the comparative statistics between the two taggers, we see that the Stanford POS tagger performs marginally better in our Twi POS tagging pipeline at 67%, 3 points higher than the NLTK tagger at 64%. One of the fallacies of the NLTK tagger was that it tagged words as a noun 79 more times than the CoreNLP tagger did, yet that only resulted in an additional nine nouns being correctly identified. Interestingly, 72% of the interjections in the dataset were categorized by the NLTK tagger as nouns.

7. Conclusion

Overall, the construction of the lemmatizer and POS tagger for Asante Twi brings into light a number of useful insights about the language. Due to the varying nature of the expression of ideas in writing (non-standardization of spellings between and within dialects, variance in the use of subject pronouns, etc), Twi data can be difficult for NLP systems to process, especially with short inputs such as single words and small phrases. In constructing the lemmatizer, it was pertinent to realize that knowledge of the language is necessary to properly lemmatize words, as rare words are often mistranslated and mistagged, and thus improperly lemmatized. A frequent example was the mistranslation of names into other parts of speech, including common nouns and adjectives. On a high level, quality of the lemmatizer is adequate; it correctly removes the prefixes and suffixes that determine tense/number. However, the method of filtration was faulty - Google Translate was not built on words [31]; it fails to correctly recognize *kɔ*, of one the most common Twi verbs, when expressed as a single word.

The better of the POS taggers, the Stanford CoreNLP system performed to a reasonable baseline accuracy of 67%. It also outscored the NLTK tagger in terms of both higher precision and recall. It should be noted that the parts of speech assigned to the ground truth labels were assigned by humans, and could be not perfectly reflective of the best way to tag the parts of speech of Asante Twi.

8. Limitations and Future Work

For building a part of speech tagger in the future, we would want to train a ML model using either a pre-existing or newly-generated parallel corpus. By training a machine learning model on paired Twi-English data as opposed to evaluating it solely on English data, we can better attribute the parts of speech between the two languages. Additionally, by operating at a sentence level, we can discern homonyms much more easily than we can at a token level, such as the different use cases of *na*, *ne*, and *no*.

Twi is not a heavily written language. Future translation work would like to use audio data and transform it into heavily annotated phonetic data. Much of the morphological changes represented in verb tenses is expressed as speech sound length, so extracting metadata from audio files would be a good way to form ’meta words’, which hold data like the tone and length of the speech sounds in a word, so that they can be properly identified, lemmatized, and translated.

9. Acknowledgements

I would like to extend a great amount of thanks to my advisor, Dr. Christiane Fellbaum, who's been an integral part in this work. I would also like to thank my Twi language course professor, Professor Hannah Essien, for her help in understanding the syntax and morphology of the language, as well as for providing the materials from which the project's text corpus was generated. Finally, I'd like to express my gratitude to my parents for their help in translating the course data for labeling purposes.

References

[1] "Akan (twi) at rutgers." [Online]. Available: https://amesall.rutgers.edu/languages/128-akan-twi

[2] "Nltk :: Natural language toolkit." [Online]. Available: https://www.nltk.org/

[3] "Part-of-speech tag sets - ibm documentation." [Online]. Available: https://www.ibm.com/docs/en/wca/3.5.0?topic=analytics-part-speech-tag-sets

[4] "Setup | cloud translation | google cloud." [Online]. Available: https://cloud.google.com/translate/docs/setup

[5] "The stanford natural language processing group - stanford log-linear part-of-speech tagger." [Online]. Available: https://nlp.stanford.edu/software/tagger.html

[6] "The stanford natural language processing group - stanford parser." [Online]. Available: https://nlp.stanford.edu/software/lex-parser.html

[7] "Twi." [Online]. Available: https://web.stanford.edu/dept/linguistics/languages/twi/Default.htm

[8] "(u+03b5) greek small letter epsilon unicode character." [Online]. Available: https://www.compart.com/en/unicode/U+03B5

[9] "(u+0511) cyrillic small letter reversed ze unicode character." [Online]. Available: https://www.compart.com/en/unicode/U+0511

[10] "(u+2184) latin small letter reversed c unicode character." [Online]. Available: https://www.compart.com/en/unicode/U+2184

[11] "Wordnet." [Online]. Available: https://wordnet.princeton.edu/

[12] "Stanford corenlp api in nltk · nltk/nltk wiki," 2019. [Online]. Available: https://github.com/nltk/nltk/wiki/Stanford-CoreNLP-API-in-NLTK

[13] M. Adjeisah *et al.*, "Twi corpus: A massively twi-to-handful languages parallel bible corpus," in *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, 2020, pp. 1043–1049. Available: https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00157

[14] P. Azunre *et al.*, "English-twi parallel corpus for machine translation," *CoRR*, vol. abs/2103.15625, 2021. Available: https://arxiv.org/abs/2103.15625

[15] C. Boitet *et al.*, "Mt on and for the web," in *Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering(NLPKE-2010)*, 2010, pp. 1–10.

[16] I. Caswell, "Google translate learns 24 new languages," May 2022. Available: https://blog.google/products/translate/24-new-languages/

[17] J. Christaller, *A grammar of the Asante and Fante language called Tshi [Chwee, Twi]*. Basel Evang. Missionary Society. Also sold, 1875. Available: https://books.google.com.gi/books?id=MugLAwAAQBAJ

[18] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018. Available: https://arxiv.org/abs/1810.04805

[19] GhanaNLP, "Ghana natural language processing." Available: https://ghananlp.org/

[20] GhanaNLP, "Khaya translator web app," 2021. Available: https://ghananlp.org/project/translator-webapp/

[21] M. Honnibal, "A good part-of-speech tagger in about 200 lines of python · explosion," 2013. Available: https://explosion.ai/blog/part-of-speech-pos-tagger-in-python

[22] M. Johnson *et al.*, "Google's multilingual neural machine translation system: Enabling zero-shot translation," Google, Tech. Rep., 2016. Available: https://arxiv.org/abs/1611.04558

[23] M. Khodak *et al.*, "Automated WordNet construction using word embeddings," in *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 12–23. Available: https://aclanthology.org/W17-1902

[24] Q. V. Le and M. Schuster, "A neural network for machine translation, at production scale," Sep 2016. Available: https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

[25] C. Metz, *Genius Makers: The Mavericks Who Brought AI to Google, Facebook, and the World*. Penguin Publishing Group, 2022. Available: https://books.google.com/books?id=JKxPEAAAQBAJ

[26] C. Newcomer, "Is google translate correct? a deep dive into google translate accuracy," 2022. Available: https://translatepress.com/is-google-translate-correct/

[27] E. K. Osam, "An introduction to the verbal and multi-verbal system of akan," 2003. Available: https://web.archive.org/web/20160303233616/http://www.ling.hf.ntnu.no/tross/osam.pdf

[28] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[29] N. Thorat, M. Schuster, and M. Johnson, "Zero-shot translation with google's multilingual neural machine translation system," Nov 2016. Available: thttps://ai.googleblog.com/2016/11/zero-shot-translation-with-googles.html

[30] K. Toutanova and C. D. Manning, “Enriching the knowledge sources used in a maximum entropy part-of-speech tagger,” in *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, ser. EMNLP ’00. USA: Association for Computational Linguistics, 2000, p. 63–70. Available: https://doi.org/10.3115/1117794.1117802

[31] B. Turovsky, “Found in translation: More accurate, fluent sentences in google translate,” Nov 2016. Available: https://blog.google/products/translate/found-translation-more-accurate-fluent-sentences-google-translate/

[32] Wikipedia contributors, “Twi — Wikipedia, the free encyclopedia,” https://en.wikipedia.org/w/index.php?title=Twi&oldid=1128823229, 2022, [Online; accessed 10-January-2023].

[33] Y. Wu *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *CoRR*, vol. abs/1609.08144, 2016. Available: http://arxiv.org/abs/1609.08144

10. Appendix

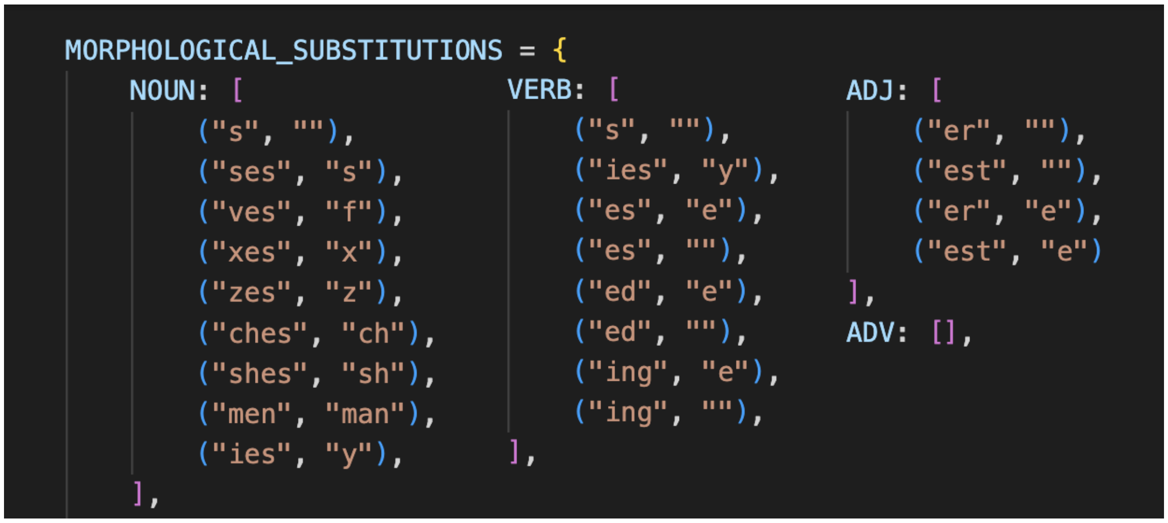
All code developed for this project is available at `https://github.com/jdapaah/Twi-Lemmatizer`.

Table 5 is an extended table detailing the morphology of verbs in Asante Twi.

Composition	Examples	Tense	English Gloss
{verb}	tɔ, da	Present	buy, sleep
/bɛ/ + {verb}	bɛtɔ, bɛda	Future	will buy, will sleep
/re/ + {verb}	retɔ, reda	Progressive	am buying, am sleeping
/rebɛ/ + {verb}	rebɛtɔ, rebɛda	Immediate Future	about to buy, about to sleep
{verb} + {X}	tɔɔ, daa	Past with complement (comp.)	bought, slept
{verb} + /eɛ/	tɔeɛ, daeɛ	Past without comp.	
/a/ + {verb}	atɔ, ada	Present Perfect (PrP)	have bought, have slept
/n/ + {verb}	ntɔ, nna	Present Negative	doesn't buy, doesn't sleep
		Future Negative	won't buy, won't sleep
/ren/ + {verb}	rentɔ, renna	Progressive Negative	am not buying, am not sleeping
		Immediate Future Negative	isn't about to buy, isn't about to sleep,
/an/ + {verb}	antɔ, anna	Past Negative	didn't buy, didn't sleep
/n/ + {verb} + {X}	ntɔɔ, nnaa	PrP Negative with comp.	haven't bought, haven't slept
/n/ + {verb} + /eɛ	ntɔeɛ, nnaeɛ	PrP Negative without comp.	

Table 5: Examples of the morphological changes to denote verb tense in Asante Twi, where {X} represents a repeat of the last letter in the root

Figure 9 lists the endings removed from words in the NLTK lemmatrization package.



```
MORPHOLOGICAL_SUBSTITUTIONS = {
    NOUN: [
        ("s", ""),
        ("ses", "s"),
        ("ves", "f"),
        ("xes", "x"),
        ("zes", "z"),
        ("ches", "ch"),
        ("shes", "sh"),
        ("men", "man"),
        ("ies", "y"),
    ],
    VERB: [
        ("s", ""),
        ("ies", "y"),
        ("es", "e"),
        ("es", ""),
        ("ed", "e"),
        ("ed", ""),
        ("ing", "e"),
        ("ing", ""),
    ],
    ADJ: [
        ("er", ""),
        ("est", ""),
        ("er", "e"),
        ("est", "e")
    ],
    ADV: [],
```

Figure 9: NLTK lemmatizer's list of morphological endings for parts of speech in English