HAWASSA UNIVERSITY

GRADUATE STUDIES DIRECTORATE

STUDY ON PART OF SPEECH TAGGING FOR WALAITA LANGUAGE USING HYBRID ALGORITHM

MSC THESIS

By: - BETELIHEM DAWIT

ADVISOR: - ANDARGACHEW MEKONNEN (Phd)

NOVEMBER 2024

HAWASSA, ETHIOPIA

HAWASSA UNIVERSITY THE INSTITUTE OF TECHNOLOGY, FACULTY OF INFORMATICS DEPARTMENT OF COMPUTER SCIENCE,

STUDY ON PART OF SPEECH TAGGING FOR WALAITA LANGUAGE USING HYBRID ALGORITHM

A Thesis submitted to the School of Graduate Studies of Hawassa University in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science

BY: - BETELIHEM DAWIT

ADVISOR: - ANDARGACHEW MEKONNEN (PHD)

NOVEMBER 2024

HAWASSA, ETHIOPIA

APPROVAL SHEET-I

This is to certify that the thesis entitled, "**Study on part of speech tagging for walaita language using hybrid algorithm**" submitted in partial fulfillment of the requirements for the degree of Master's with specialization in Computer Science, the Graduate Program of the Faculty of Informatics, and has been carried out by **Betelihem Dawit**. Therefore we recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the department.

Andargachew Mekonnen (PhD)	_____	28/11/2024
Name of major advisor	Signature	Date
_____	_____	_____
Name of co-advisor	Signature	Date

EXAMINERS' APPROVAL SHEET

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY EXAMINERS' APPROVAL SHEET

(Submission Sheet-2)

We, the undersigned, members of the Board of Examiners of the final open defense by **Betelihem Dawit** have read and evaluated his/her thesis entitled "**Study on Part of Speech Tagging for Walaita Language using Hybrid Algorithm**", and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirements for the degree.

Andargachew Mekonnen (PhD)	_____	28/11/2024
Name of Major Advisor	Signature	Date
Dr. Degif Teka	_____	_____
Name of Internal Examiner-I	Signature	Date
_____	_____	_____
Name of Internal Examiner-II	Signature	Date
Dr. Asrat Mulatu	_____	29/11/2024
Name of External examiner	Signature	Date
_____	_____	_____
SGS Approval	Signature	Date

Final approval and acceptance of the thesis is contingent upon the submission of the final copy of the thesis to the School of Graduate Studies (SGS) through the Department/School Graduate Committee (DGC/SGC) of the candidate's department.

Stamp of SGS Date: _____

Declaration

I, hereby confirm that this thesis is solely my own work. I have adhered to ethical standards throughout the research, data collection, analysis, and writing processes. Proper citation has been provided for all scholarly content included in the thesis. I assure that all sources utilized in this work have been appropriately acknowledged. Diligent measures have been taken to prevent any form of plagiarism in the development of this thesis.

This thesis has been submitted to the Graduate Studies Directorate of Hawassa University as a partial fulfillment of the requirements for a Master degree. It has been deposited in the University Library and is accessible to borrowers in accordance with the library's regulations.

Short excerpts from this thesis can be utilized without specific permission as long as proper and full credit is given to the source. For longer quotations or reproduction of the thesis, authorization may be required from the appropriate academic authorities. Permission from the author is necessary for any other use of the thesis material.

Declared by:-

Name of student	Signature	Date
Betelihem Dawit	_____	_____

This research has been submitted for Examination with my approval as University advisor.

Name of Advisor	Signature	Date
Andargachew Mekonnen (PhD)	_____	28/11/2024

Acknowledgments

First and foremost, I would like to express my gratitude to the Almighty God for making this endeavor a reality and for guiding me to achieve this level of success, despite the numerous challenges I faced along the way.

Additionally, I would like to extend my heartfelt appreciation to my advisor, Dr. Andargachew Mekonin, for his fascinating guidance, unwavering support, and invaluable suggestions, which played a vital role in completing my thesis study. Lastly, I am immensely grateful to all my teachers who have supported me, provided valuable feedback, and encouraged me throughout my postgraduate journey, from the beginning of my classes until the culmination of this thesis project. I sincerely thank Wolaita Sodo University Wolaita Language department Teachers for their support and cooperation during the data collection and brainstorming session.

Finally, I would like to express my sincere gratitude and deepest love to my beloved family for their daily motivation and for supporting me at every step. I would also like to thank all my friends for their support and encouragement.

ABSTRACT

Natural Language Processing is a field of computer science, artificial intelligence, and computational linguistics concerned with the interaction between computers and human (natural) languages. Part-of-speech tagging is the process of automatic annotation of lexical categories. Part-of-speech tagging assigns an appropriate part-of-speech tag for each word in a sentence of natural languages.

The parts-of-speech (PoS) tagging for Wolaita language is not sufficient yet to be used as one important component in other natural language processing (NLP) applications.

In this research, the part of speech tagger for the Wolaita Language is based on a hybrid approach; it combines the Rule-based method with an HMM model. As we have mentioned, the Rule-based method is composed of three steps: lexicon analyzer, morphological analyzer, and syntax analyzer. Almost all words are recognized by the rule-based method. However, some terms are not analyzed or misclassified. These terms (the rest failed terms) will be analyzed using the HMM model.

For implementation and experiment, the author used python programming and NLTK. For tagging purpose we used 17 PoS tag sets. For training and testing the model, 79,925 Wolaita language words are collected from three different categories. From the entire corpus, 90% is used for training and the remaining 10% is used for testing purpose. The performance of the tagger, are tested by using different portion training dataset using percentage split method. Accordingly, after correcting human and spelling errors, the performance of the tagger shows 66069 words (95.77%) correctly tagged words for Rule Based and 2319 words (99.2%) correctly tagged words for HMM model in the Train_ Set. The performance of the tagger shows 6955 words (90.5%) correctly tagged words for Rule Based and 277 words (98.4%) correctly tagged words for HMM model in the Test-set. As a result, the results of the experiments for both Rule bases and HMM algorithms show an accuracy of 93.13% and 98.3% correctly tagged words on average respectively. Increasing the size of training data and examining the tagger influences the result. Result from our experiment shows that adding of HMM tagger performs better result than rule based tagger alone.

Keywords: Wolaita Language, part-of-speech tagger, Natural Language Processing, Hidden Markov Model, Corpus

Table of Contents **Page Numbers**

List of Tables

List of Figures

List of Acronyms and Abbreviations

- **ANN:** Artificial Neural Network
- **HMM:** Hidden Markov Model
- **LexProb:** Lexical Probabilities
- **LOB** : Lancaster-Oslo-Bergen
- **NLP**: Natural Language Processing
- **NLTK:** Natural Language Processing Toolkit
- **POS**: Part of Speech
- **POST**: Part of Speech Tagger
- **TBL:** Transformation Based Learning
- **TEL:** Transformation-based Error-Driven Learning
- **TransProb:** Transitional Probabilities
- **TTS:** Text To Speech
- **WBS:** Work Breakdown Structure

CHAPTER ONE: INTRODUCTION

1.1. Background of the Study

Wolaita or Wolaita Language is a North Omotic language of the Ometo group spoken in the Wolaita zone and other parts of southern nations, nationalities, and the People's Region of Ethiopia or South Ethiopia Region. It is the native language of the Wolaita people. The estimation of the population varies greatly because it isn't agreed upon where the boundaries of the language are [1].

The idea of computers being able to understand ordinary languages and hold conversations with human beings has been a staple of science fiction since the first half of the twentieth century and was envisaged in a classic paper by Alan Turing (1950) as a hallmark of computational intelligence [2].

Natural Language Processing is a field of computer science, artificial intelligence, and computational linguistics concerned with the interaction between computers and human (natural) languages, and in particular, with programming computers to process large natural language corpora[3] fruitfully.

Nowadays, contributions to the natural language processing (NLP) field have been enhanced significantly, which has allowed for the processing of large amounts of textual data with an acceptable level of importance. The application of natural language processing techniques as components in web search engines, machine translation tools, and summary generators are some of the instances of textual information processing [4]. Machine translation, grammar checking, information extraction and retrieval using natural language, automatic written text recognition, text-to-speech synthesis, and part of speech tagging are some of the areas of natural language under research and investigation [5]. Languages like English [5], [6], Turkish[7], and Ethiopian languages such as Amharic[8], Afaan Oromo[9], [10], Tigrign[11], and others are among the languages where natural language applications are being developed.

Part-of-speech (POS) tagging is the process of automatic annotation of lexical categories. POS tagging assigns an appropriate part-of-speech tag for each word in a sentence of natural language. POS tags are also known as word classes, morphological classes, or lexical tags. POS tagging can be used in text-to-speech (TTS) applications, information retrieval, parsing, information extraction, and linguistic research for corpora [12], [13] and it can also be used as an intermediate step for higher-level NLP tasks such as parsing, semantics analysis, translation, and many more[6], which

makes POS tagging a necessary function for advanced NLP applications in Wolaita or any other language.

There are three general approaches to dealing with the tagging problem: a rule-based approach, a statistical approach, and a hybrid approach. The rule-based approach consists of developing a rule-based knowledge base established by linguists to define precisely how and where to assign the various POS tags. The statistical approach consists of building a trainable model and using a previously-tagged corpus to estimate its parameters. Once this is done, the model can be used to determine the tags of other texts. Generally, successful statistical taggers are mainly based on hidden Markov models (HMMs). Finally, the hybrid approach consists of combining a rule-based approach with a statistical one. Recently, most of the POS taggers have used the latter approach, as it gives better results. However, in this study, the hybrid approach (rule-based and statistically hidden Markov model) is used to develop part of the speech tagger for the Wolaita language.

1.2.Statement of the problem

As far as this research on NLP is concerned, few types of research have been done in the area of Wolaita-language processing. There are parts of speech taggers developed for local languages like Amharic [14], [8]. The previous researcher studied the POS tagging for the Wolaita language, and she used a few words from Corpus [15]. Research in areas of POS tagging will contribute a lot to the effort of natural language processing in the Wolaita language. The absence of the POS tagger system limits research concerning the NLP of the Wolaita language, such as parsing (syntactic and semantic), machine translation, sentence grammar checking, spell checking, speech synthesis, etc. It is used as a pre-processing component for the aforementioned NLP applications. Hence, conducting research and developing an automatic part of speech tagger for the Wolaita- language is of supreme significance.

1.3.Objectives

1.3.1. General Objectives

The general objective of this research is to design and develop a POS tagger for the Wolaita-language.

1.3.2. Specific objectives

To achieve the above General Objective, the research accomplished the following specific objectives:

- To review the basic word category of the Wolaita language for identifying the POS tags that will help identify the tag sets for the language.
- To collect and design a corpus for training and testing of the system
- To assess the various approaches for the development of POS Tagger for the language.
- To develop and test a prototype tagger and analyze its performance for the Wolaita language POS tagger.
- To draw conclusions based on experimental results and recommend further research areas in the future.

1.4.Research Questions

To achieve its Objectives: the Study raised the following research questions:

1. What are the possible tag sets for the Wolaita-language?
2. How to develop a corpus for the Wolaita- language?
3. What are the different approaches that can be used to develop part of the speech tagger for the Wolaita language?
4. How is the design part of a speech tagger model using a specific algorithm?

1.5.Scope of the Study

The main goal of this proposed research is to design and develop POS Tagger for the Wolaita language.

The corpus developed in this research work is a domain-specific corpus, a text corpus that is collected from a different domain, in this case, the news domain, the holy book, student books, and other domains. Moreover, during the development of the corpus, the tag sets are meant to give information about words, their word class category, and the issues.

Therefore, the research studied will be up to 50,000 words. More than that, it requires more time to create a corpus and may include a shortage of time, budget, and support. The basic problem for this

proposed research work was the lack of the data corpus needed for building a model and testing and the lack of language experts.

1.6. Significance of the Study

This research study aims to identify the weaknesses and challenges of using the Wolaita language and to develop taggers, or a class of words, for the Wolaita language. There are many advantages to developing a POS tagger for a specific language. It is the basis for developing other higher-level applications of NLP such as parsing, information extraction, information retrieval, question-answering, spelling correction, machine translation, etc. This application can be used in different areas of the Wolaita-language. Accordingly, the benefits of this study are:

- Who wants to conduct the higher-level application of NLP for this language, such as parsing, spell checking, grammar checking, speech recognition, question answering, etc.?
- People who want to learn the Wolaita Language as a second language may find it helpful to discover the word categories and grammar construction.
- It serves as a reference for those who want to engage in a study on the same issues.

1.7. Thesis organization

This thesis is intended to have five chapters.

Chapter one has presented the general background of the study, the statement of the problem, objectives, significance, and scope of the study. Chapter two presented a literature review on the part of speech tagging. Chapter three has presented the Research Methodology. Chapter four describes Results and Discussions. Chapter Five presents the conclusion and recommendations for future works.

CHAPTER TWO: LITERATURE REVIEW AND RELATED WORKS

2. INTRODUCTION

This chapter tries to ascertain an understanding of the literature and related works that have been conducted in the area of the part of speech tagger development both locally and globally.

2.1. Literature review

Part of Speech tagging is the process of identifying the part of speech corresponding to each word in the text based on both its definitions. As well as its context (i.e. relationship with adjacent and related words in a phrase or sentence [4], [2], [5] .

E.g. if we consider the sentence 'The white dog ate the biscuits' we have the following tags

The [DT] white [JJ] dog [NN] ate [VBD] the [DT] biscuits [NNS]

Several approaches have been proposed to annotate words automatically with their part-of-speech tags. Namely Rule-Based Part of Speech Taggers, Stochastic Part of Speech Taggers, ANN Approach, and Hybrid Approach.

2.1.1 Rule-Based Approach

Rule-Based Tagging: Algorithm for assigning part of speech based on two stages, the first stage uses a lexicon (dictionary) to assign each word a list of potential parts of speech and the second stage uses a huge list of handwritten disambiguation rules to winnow down this list to a single part of speech for each word [16]. Contextual information is used for assigning tags to unknown words in a typical rule-based approach. The rules are usually said to be context frame rules something like: "If an ambiguous/unknown word X is preceded by a determiner and followed by a noun, tag it as an adjective det - X - n = X/adj Can be considered as an example of context frame rules[9].

Together with contextual information, many taggers apply morphological information to assist the disambiguation process something like: "If an ambiguous/unknown word ends in an-ing and is preceded by a verb, label it a verb" is one example of such rules. There are still systems that not only use contextual and morphological information but also include other rules such as capitalization and punctuation. The language being tagged determines the value of information of

this type to be greater or lesser. Capitalization for example proves highly useful in the tagging of unknown nouns in the German language [17].

Usually supervised training is required for rule-based taggers [16]. But nowadays there has been a growing deal of desire for the automatic generation of rules. Running an untagged text through a tagger and seeing how it performs is among the methods of automatic rule generation.

The output of the first phase is analyzed by a human being and then corrections are made if there are any erroneously made tagged words. The correctly tagged text is given to the tagger for learning correction rules by comparing the two sets of data. This process is usually repeated several times. Klein and Simmons are pioneers of this approach. Avoiding the labor of constructing a huge dictionary was their major goal. The algorithm they developed uses 30 different tag sets or POS categories. Seeking each word in the dictionary followed by checking for suffixes and special characters as clues and finally applying context frame tests are the steps of this algorithm. These work on scopes bounded by unambiguous words. However, Klein and Simmons impose an explicit limit of three ambiguous words in a row. For each such span of ambiguous words, the pair of unambiguous categories bounding it is mapped into a list. The list includes all known sequences of tags occurring between the particular bounding tags; all such sequences of the correct length become candidates. The program then matches the candidate sequences against the ambiguities remaining from earlier steps of the algorithm. When only one sequence is possible, disambiguation is successful. This algorithm correctly and unambiguously tags about 90% of the words in several pages of the Golden Book Encyclopedia [12], [6], [18].

TAGGIT was developed by Green and Rubin in 1971 [12]. Compared with Klein and Simmons large tag sets were used about 86 tag sets. The tagged Brown corpus is the source of the dictionary used than the untagged version [12]. The task of category assignment is divided into initial tagging and disambiguation by tagging. According to [17] Tagging is carried out as follows; first, the program consults an exceptional dictionary of about 3,000 words. Among other items, this contains all known closed-class words. It then handles various special cases, such as words with the initial "$", contractions, special symbols, and capitalized words. A word's ending is then checked against a suffix list of about 450 strings that were derived from the Brown Corpus. If TAGGIT has not assigned some tag(s) after these several steps, the word is

tagged as a noun, a verb, and an adjective, so that the disambiguation routine may have something to work with. This tagger correctly tags approximately 77% of the million words in the Brown Corpus (the rest is completed by human post-editors) [12]. A very successful constraint-based approach for morphological disambiguation, known as Constraint Grammar, was developed in Finland, from 1989 to 1992, by four researchers: Fred Karlsson, Arto Anttila, Juha Heikkila, and Atro Voutilainen. In this framework, the problem of parsing was broken into seven sub-problems or modules; four of them are related to morphological disambiguation, and the rest are used for parsing the running text. One of the most important steps of Constraint Grammar was context-dependent morphological disambiguation, where ambiguity is resolved using some context-dependent constraints. For this purpose, they wrote a grammar, which contains a set of constraints based on descriptive grammar and studies of various corpora. Each constraint is a quadruple consisting of domain, operator, target, and context condition(s) [19].

The recent rule-based algorithm is transformation-based tagging (TBL); it is commonly known as Brill's rule-based tagger [20]. Unlike earlier algorithms, it uses a supervised learning technique that uses a pre-manually tagged training corpus. Brill's TBL algorithm has also stages to assign a word with the appropriate tag. In the first stage, each word is assigned its appropriate lexical category from annotated corpora. As a preprocessor, it assigns the most likely (frequent) tag to each word; for example, he/pro shall/modal go/verb a/art house/noun.

Brill's tagger then applies a list of transformations to alter the initial tagging. Transformations are contextual rules that rewrite a word tag into a new one. The transformation is performed only if the new tag of the word is legal, in the dictionary. If so, the word is assigned the new tag. Transformations are executed sequentially and each transformation is applied to the text from left to right [21]. Changing the modal to a noun is applied as a rule for the English language if the previous word is an article. The rule is applied to a sentence, *the/art can/noun rusted/verb*. Brill's rules tagger conforms to a limited number of transformation types, called templates [22]. For example, the rule changes the tag from modal to noun if the previous word is an article, and corresponds to the template. Table 1 shows a sample template that is used in Brill's rule tagger. Where, A, B, and C represent lexical categories or parts of speeches.

Table 1: Sample Template in Brill's Rule.

Rules	Explanation
alter(A, B, prev tag(C))	Change A to B if the preceding tag is C
alter(A, B, next tag(C))	Change A to B if the following tag is C

Adding a rule to a system may involve over-generation, i.e., one extra rule can result in more harm to the accuracy of general tagging in a machine-learning rule-based approach. The manual rule-based tagging system has also the aforementioned limitations [8]. Therefore, in general, rule-based approaches are time-consuming and require a great knowledge of a specific language being tagged. But it is also possible to find some advantages of the rule-based approach that is listed in the work of [16]. These are Robustness, a vast reduction in stored information required by the lucidity of a small set of meaningful rules, ease of finding and implementing improvements to the tagger, and better portability from one tag set or corpus type to another. According to [11] generally, the basic characteristics of the rule-based approach are clarified as advantages and disadvantages below:

Advantages:

- requires only a small amount of training data
- useful for a limited domain
- Can be used with both well-formed and ill-formed input
- High quality based on solid linguistic

Disadvantage:

- Most of the time, it relies on hand-constructed rules that are to be acquired from language specialists and the construction of these rules is tedious and time-consuming.
- development could be very time-consuming
- not easy to obtain high coverage of linguistic knowledge
- some changes may be hard to accommodate

2.1.2 Statistical (Stochastic) Approaches

The term 'stochastic tagger' can refer to any number of different approaches to the problem of POS tagging [21]. Any model, that somehow incorporates frequency or probability, i.e. statistics,

may be properly labeled stochastic. The simplest stochastic taggers disambiguate words based solely on the probability that a word occurs with a particular tag. In other words, the tag encountered most frequently in the training set is the one assigned to an ambiguous instance of that word.

According to [23] the problem with this approach is that while it may yield a valid tag for a given word, it can also yield inadmissible sequences of tags. An alternative to the word frequency approach is to calculate the probability of a given sequence of tags occurring. This is sometimes referred to as the n-gram approach, referring to the fact that the best tag for a given word is determined by the probability that it occurs with the n previous tags [24]. The most common algorithm for implementing an n-gram approach is known as the Viterbi Algorithm, a search algorithm that avoids the polynomial expansion of a breadth-first search by "trimming" the search tree at each level using the best N Maximum Likelihood Estimates (where n represents the number of tags of the following word) [16]. If n is one (i.e. 1-gram approach), it is just the word frequency. If n is two (2-gram approach), it has a special name: bi-gram approach. For example, if you consider the frequency of “the smile”, where “the” is the determiner and “smile” is a common noun, this approach is a bi-gram approach. Similarly, if you consider the frequency of n words, ordered, it is an n-gram approach. The next level of complexity that can be introduced into a stochastic tagger is the one that combines the previous two approaches, using both tag sequence probabilities and word frequency measurements.

Supervised and unsupervised taggers are the two classifications of a stochastic approach based on the training data the tagger uses. Supervised taggers use a corpus where a word class is attached to every token [23] which is used for training to learn information about tag set and word frequencies. On the other hand, an unsupervised tagger uses a Baum-Welch algorithm to learn information about the tag set and word frequency from a n untagged corpus[24], [23]. This algorithm is used when no pre-tagged corpus exists for training purposes. This approach can be based on different methods n-gram, maximum-likelihood, and HMM [23].

Maximum-Likelihood (Most Frequent Tag) Method

The most frequent tag model as its name implies tries to pick the most frequent tag for a given word in a sentence. This model is the simplest in the stochastic approach as it simply finds the most frequent tag from the training corpus [23]. This can be done by counting the occurrence of the specific word Wi associated with a tag Ti and dividing it by the total occurrence of the word

Wi in the training corpus which can be represented mathematically as:

$$P(Ti|Wi) = \frac{count\ of\ (Ti, Wi)}{count\ of\ (Wi)}$$

This implies that the most frequent tag model computes the probabilities of observing each word attached to every part of speech tags during the training phase. Then in the tagging process of new text, it will pick the tag with the most probable tag for that word. This model has very clear limitations as it does not try to look into the sentence structure. It rather picks the most frequent tag for a given word [9].

N-gram Method

The N-gram model is a mechanism for dealing local context of words in a given sentence. It is originally conceived as a technique for predicting the next element of a sequence given only the N-1 previous elements [25]. The elements can be a sequence of words, tags, or both words and their corresponding tags. This implies that it can be used for finding the tag sequence probabilities (probability of a tag Ti given the previous N-1 tags (P(Ti|Ti-1, Ti-2, …, Tn)) or word sequence probabilities ((P(Wi|Wi-1, Wi-2, …, Wn)). Moreover, the n-gram can also be used for finding the probability of the tag of a current word given the previous n words (P(Ti, Wi |Wi-1, Wi-2, …, Wn)). It solves the problem of the most frequent tag model which ignores the local context of words. This model decides the appropriate tag for a word by computing the probability that it occurs within the n-previous tags, where the value of N is considered to be 1, 2, or 3 for practical purposes. These are known as unigram, bigram, and trigram models respectively [22].

HMM Method

The Hidden Markov Model (HMM) is the most widely used model for part-of-speech tagging under the stochastic approach [21]. The main idea of the HMM is to find the sequence of tags for a given sequence of words. This can be done by combining the most frequent tag and the N-gram model that considers the tag sequence probabilities i.e. considering the lexical category of a word using its most frequent tag and its local context in the sentence from the training corpus during the training phase. Generally speaking, from a statistical point of view, the task of the HMM model is to find the most likely POS sequence T1, T2, … Tn for a given word sequence W1, W2,

…Wn. In other words, the model has to maximize the conditional probability of the tag sequence given the word sequence over all possible tag sequences Tn [7]. Putting it all together, this approach tries to maximize the probability P (T1, T2, ..., Tn …… Tn | W1, W2, ..., Wn) which can be achieved with the help of the N-gram model and most frequent tag model. HMM is the probabilistic function of the Markov Process, a process that moves from state to state, from left to right on the states, to find the optimal state sequence. An HMM is characterized by the following criteria [16]:

- A finite set of states each of which is associated with a probability distribution
- Transitions among the states are governed by a set of probabilities called transition probabilities.
- In a particular state, an outcome or observation can be generated according to the associated probability distribution. The observation is visible and the states are hidden from the observer hence the name Hidden Markov Model.

HMM is defined formally as a set {S, O, A, B} Where [16]:

- S denotes the set of states
- O denotes the set of observation symbols.
- A = $\{a_{ij}\}$ is a set of state transition probabilities represented in a transition probability matrix in which each represents a probability of moving from state S_i at time t into state S_j at time t+1.

Generally, an HMM is a set containing {S, O} where:

$S = \{S_1\ S_2\ S_3\ S_4,\ldots\ldots, S_n)$

$O = \{O_1\ O_2\ O_3\ O_4,\ldots\ldots,O_m)$

= {A, B}

HMM considers three assumptions [9]. The first assumption is called the Markov assumption which states the first-order transition probability can be extended to the k^{th}-order transition probability. It implies that, if it is possible to find the probability of state S_i given the previous state S_{i-1} ($P(S_i|S_{i-1})$), it would be also possible to find the probability of state S_i given the previous K states ($P(S_i|S_1, S_2, \ldots S_k)$. The second assumption is called the stationary

assumption that says: state transition probabilities are independent of time and the output. It takes due consideration of the actual time at which the state transition takes place and the output symbol that can be emitted being on the particular state [26]. As a result, it considers that state transition probabilities are independent of the actual time and output symbol, which can be represented mathematically as:

$P(S_{t1+1}|S_{t1}) = P(S_{t2+1}|S_{t2})$ for any time t_1 and t_2.

The third assumption is the output independence assumption that states: the current observation is statistically independent of the previous observations. This means the probability of observing an output symbol O_i being in state S_i is independent of the probability of observing O_{i-1} being in state S_i. This can be represented mathematically as $P(O_i|S_1, S_2, S_3 \ldots S_n)$. Because of these assumptions, HMM fails to accurately find the most likely sequence of states for a given sequence of observations.

When the HMM model is taken to the application of POS Tagging, the hidden states are the POS tags (tag sets) and the sequences of words are the sequence of observations. The transition probability in POS tagging is the probability of moving from one tag to the next tag and the emission probability is the probability of getting a word W_i being in tag T_i. The main problem of the HMM from the POS tagging point of view is, finding the sequence of tags thereby maximizing their probabilities given a sequence of words in a text. This can be done using the Viterbi Algorithm which will take the joint probabilities of the lexical probability and the n-gram probability [19].

According to [9] the characteristics of the stochastic method are listed below as advantages and disadvantages:

Advantages:

- Researchers may not need language specialists, expertise
- Coverage depends on the training data

Disadvantages:

- requires a large amount of annotated training data (very large corpora)
- some changes may require re-annotation of the entire training corpus in the supervised statistical learning

- Not easy to work with ill-formed input as both well-formed and ill-formed are still probable

2.1.3 ANN Approach

An artificial neural network is a system that is composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strength, and the processing performance at the computing element or node [27]. The other definition of the artificial neural network is given by [28] which define it as a massively parallel distributed processor that has a natural way of storing learned knowledge and making it available for use.

It resembles the human brain in two aspects:

1. Knowledge is acquired by the network through the learning process.
2. Inter-neuron connection strength known as synaptic weights is used to store knowledge.

Though the field of Artificial Neural Networks was established before the advent of computers, the ANN simulations appear to be a recent development. This implies that the information processing of ANN was known in the biological nervous system before actually applying it in information processing using computer applications [11]. Hence, the basic idea of ANN here is to process information in a similar way to the biological nervous system process pattern [11]. The key element of the ANN information processing scheme is the novel structure of the information processing system which is composed of, like the biological nervous system, a large number of highly interconnected elements called neurons working in union to solve a specific problem [29], [10], [8]. Artificial Neural Networks learn from examples by configuring for a specific application such as pattern recognition or data classification. The ANN can learn by adapting different behaviors based on the data that is given to the network. It is possible to call the ANN learning adaptive learning as the network can find properties from the presented data. It is not necessary to tell the network how to react to each data input separately like the conventional programming.

The common types of ANNs consist of three layers of units namely a layer of input units, a layer of hidden units, and a layer of output units [8], [11]. The input layer which is connected to the hidden layer represents the raw information that is fed to the network as an input so that it can learn and adapt properties. The middle layer, the so-called hidden layer, connected to the output layer is determined by the activities of the input unit and the weights on the connections

between the input and hidden units. The output layer represents the result of the learning properties from the input layer and hidden layer.

Taking the ANN approach to the application of part of speech tagging, first preprocessing activities are performed before dealing actually with the ANN-based part of speech tagger. Such preprocessing activities can be tokenizing, and feature extraction like POS information, word information, POS category, and order information, etc. The result of the preprocessing activities is given to the input layer of the network from which the network can learn patterns. As mentioned above, the input layer is connected to the hidden layer, and in this layer, different algorithms like error back-propagation algorithm, an algorithm based on an error-correction learning rule specifically on the minimization of the mean squared error which is a measure of the difference between the actual and the desired output, can be used for training the system [11].

This technique of tackling the problem of assigning part of speech tags to words has some disadvantages compared to the HMM and rule-based approaches. Some of these are: The HMM method assigns the sequence of tags for the sequence of words in the entire sentence i.e. it takes the due consideration of the sentence structure while most ANNs take the word to be tagged only. The same thing is true with rule-based approaches which tend to take the sentence structure and generally the linguistic patterns into consideration [27].

2.1.4 Hybrid Approach

As its name implies, this approach takes the combination of either stochastic approach and rule-based approach or ANN and rule-based by taking advantage of both approaches to improve the performance of the system. This approach combines either a rule-based and probabilistic approach or a rule-based and artificial neural network approach to enhance the performance of the part-of-speech tagger. Some researchers used a hybrid approach (stochastic and rule-based approach) [7], [30], [11]. Others implement a hybrid approach that combines an artificial neural network and a rule-based approach [8].

2.2. Related works

2.2.1. Previous Work on Local Languages

Birhanesh Fikre Shirko, (2020), Part of Speech Tagging for Wolaita Language using Transformation Based Learning (TBL) Approach. In their work, they used 1020 sentences or

12,683 words for the training dataset and 114 Sentence or 1,675 words for the testing dataset, and the total, preprocessed data for the study is 1134 Sentences or 14,358 words to develop PoS Tagging for the Wolaita language [15].

Getachew Mamo and Million Meshesha adopted the HMM approach to develop the earliest part of the speech tagger for the Afan Oromo language. In their work, they used 159 sentences for training and testing purposes which have around 1621 tokens collected from different sources to balance the corpus [9].

The tagger assigns lexical classes to Afan Oromo text in two major phases. The first phase begins by training the tagger in the training data for computing and storing both the lexical and contextual probability of training data. In the second phase, the tagger receives untagged Afaan Oromo text and tokenized into words. Then, the tagger assigns the correct POS tag for each token. This is achieved by using the unigram and bigram model of the Viterbi algorithm by taking the stored information during the first phase.

The researchers have tested the performance of the tagger using a tenfold cross-validation mechanism. As a result, they got 87.58% and 91.97% accuracy for the unigram and bigram models respectively.

Another work on the Afaan Oromo language part-of-speech tagger was done by Mohamed Hussein [10]. In this work, the researcher adopted transformation-based tagging (Brill tagging), with some modifications to the original tagging template. This approach is an instance of the transformation-based learning (TBL) approach to machine learning [16], and it draws inspiration from both the rule-based tagger and stochastic taggers. Like the rule-based taggers, it is based on rules that specify which tag should be assigned to a word. But like stochastic taggers, it is a machine-learning technique in which rules are automatically induced from the training data. The researcher used 233 sentences (1708 words) from different sources to make the corpus balanced. Out of this, 90% of the corpus was used for training purposes while the remaining 10% was used for testing. In addition to this, he has identified 18 tag sets.

To learn a set of transformation rules, a TEL (Transformation-based Error Driven Learning) based tagger takes an annotated corpus as input and goes through the initial-state tagger, that assigns the most likely tag. This gives a temporary output. The temporary corpus is compared with

the one that is manually tagged and assumed to be correct. Based on the result of the comparison, the temporary outputs pass through both lexical and contextual learners to drive the rule of transformations. Each transformation rule is tested on the temporary output to select the best transformation rule that maximizes the performance of the tagger. The rule with the best score is applied to the temporary corpus to produce the next temporary corpus and added to the set of rules. Starting from the comparison of the temporary corpus with reference text to transformation rule generation, the process continues in the same fashion to produce all the permissible rules with the corresponding temporary text until no rule can further improve the tagging of the temporary corpus. Afterward, the tagger accepts unannotated Afaan Oromo text and assigns the best tag for each word based on the rules that are learned. To test the performance of the proposed method and original Brill tagger, the researcher has conducted ten experiments on different numbers of words. As a result, he has got an average accuracy of 80.80% and 77.64% for modified and original Brill taggers respectively.

A hybrid approach by applying a combination of rule-based and statistical approaches has been introduced for the Tigrigna language part-of-speech tagger by Teklay G/Egzabiher [11]. In this work, the researcher used a combination of HMM, which is widely used under the stochastic approach, and adapting the Brill transformation-error-driven learning approach to drive machine-learned rules for designing the rule-based tagger component. The researcher has collected a total of 26,000 words from Tigrigna news broadcasting agencies and annotated them manually with their corresponding word class. In addition to this, he has identified 36 tag sets for the entire tagging process. Among the total words, 75% (20,000) words used for training purposes while the remaining 25% (6000) words used for testing purposes.

Generally, this study finds the tag of a word in two main steps. The first step is performed by the HMM tagger. The HMM tagger first annotates the given raw text and provides a level of confidence (threshold value) for each tag sequence. In the second step, the confidence level of each tag sequence is compared with the minimum confidence level that is set by the researcher using the output analyzer module. If the confidence level is less than that of the minimum confidence level, a window size of two (bigram of the word) is given to the rule-based tagger for correction. Otherwise, it is treated as a correct tag. To test the accuracy of the proposed method, the researcher conducted different experiments for the three types of taggers namely HMM tagger, rule-based tagger, and hybrid tagger. As a result, he has an accuracy of

89.13% for HMM, 91.8% for rule-based, and 95.88% for hybrid tagger.

2.2.2. Previous Work on Foreign Languages

In 1983, Marshall described the Lancaster-Oslo-Bergen (LOB) Corpus tagging algorithm, later named CLAWS. It is similar to the TAGGIT program [12]. The tag set used is very similar, but somewhat larger, at about 130 tags. The dictionary used is derived from the tagged Brown Corpus, rather than from the untagged version. The main innovation of the CLAWS is the use of a matrix of collocation probabilities, indicating the relative likelihood of co-occurrence of all ordered pairs of tags. This matrix can be mechanically derived from any pre-tagged corpus. CLAWS used a large portion of the Brown Corpus, with 200,000 words. CLAWS have been applied to the entire LOB Corpus with an accuracy of between 96% and 97%.

Eric Brill [16], [17] proposed another approach to POS tagger called Brill tagger. It is a kind of transformation-based learning named after its inventor. He proposed a system that guesses the tag of each word, then goes back and fixes the mistakes. The general idea of the tagger is to assign each word within a given text it's most likely tag estimated by an initial-state tagger that is trained on a large tagged corpus without regard to context [16]. Once the text passes through the initial state tagger, it is compared with the reference text (manually tagged text). As a result, an ordered list of transformation rules is learned that can be applied to the output of the initial-state tagger to make it better resembles the reference text.

To test the performance of the system, the researcher conducted an experiment using 1.1 million words of the Pen Tree-bank tagged Wall Street Journal corpus. From these, 950,000 words were used for training, and 150,000 words were used for testing. Out of 950,000 words of the training corpus, 350,000 words were used to learn rules for tagging unknown words, and 600,000 words were used to learn contextual rules. As a result, the experimental performance of this approach indicates 96.6% accuracy on the test corpus.

A composite POS tagger for Turkish, which combines a rule-based and statistical approach, was developed by Levent Altanyart, Zihni Orhan, and Tunga Gunger [7]. In this work, the researchers used both word frequencies and n-gram (unigram, bigram, and trigram) probabilities. To increase the performance of the system, the researchers use two additional features. In the first case, they incorporate a morphological analyzer which is used to obtain the part-of-speech of words independent of the words within the corpora. This enables the system

to guess the tag of the word even if it does not exist within the corpus. In the second case, the researchers use another statistical approach which is related to the part-of-speech of words based on the position within the sentence.

To develop the system, the researchers have collected 7200 sentences. Among the 7200 sentences, 6000 (85%) of the corpus was used for training and the remaining 1200 (15%) of the sentences were used for testing purposes. In addition to this, they identified 13 word classes for the tagging process.

The system finds the tag of a word in three main steps. In the first step, the statistical analyzer module computes some statistical data from the training corpus. In the second step, the tag set finder finds possible parts of speech for words to be tagged. Finally, the main modules of the system determine part-of-speech of words. To reach the final decision the tagger combines word frequencies, n-gram probabilities, heuristics data, and data about candidate tags.

To test the performance of the system, with and without statistical data, the researchers have performed three experiments by using different parts of the corpus as training and testing sets. As a result, they have an average accuracy of 82.26% for systems with statistical data and 66.73% for systems without statistical data.

2.3. LINGUISTIC PROPERTIES OF WOLAITA LANGUAGE

Wolaita language is an Omotic language spoken in the Wolaita Zone and some other parts of South Ethiopia. The number of speakers of this language is estimated at 2,000,000 (1991 UBS); it is the native language of the Wolaita people [31]. The natives call the language of Wolaita Language, which means the language of Wolaita.

Wolaita Language has existed in written form since the 1940s when the Sudan Interior Mission first devised a system for writing it. The writing system was later revised by a team led by Dr. Bruce Adams. They finished the New Testament in 1981 and the entire Bible in 2002 [32].

In some institutions like Wolaita Sodo University (WSU), there is the Wolaita Language department that offers this language as a subject. It is also thought of as a subject in the junior and secondary schools of the Wolaita zone.

Wolaita Language has 34 Letters which are used to form words. These are written in two ways

capital and small letters.

Table 2: Wolaita Language letters

Pittaliya/ Capital Letters	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z CH DH NH NY PH SH TS ZH
Pittaliya/ small Letters	a b c d e f g h i j k l m n o p q r s t u v w x y z ch dh nh ny ph sh ts zh

These thirty-four letters are divided into vowels and consonants. In Wolayta language there are five different vowels. These are: - a, e, i, o, u. these letters are written in two ways. Shortening and lengthening. When the word is short, we write a single letter like:

“a", "e", "i", "o", and "u" and for the long words we write the double vowels capital like: “aa”, “ee", "ii", “oo” and “uu”.

Table 3: Wolaita Language letters shortening and lengthening usage example.

Qaamiyode/Shortening	Aduqqiyode/Lengthening
Akkaa/dew	Aaka/be wide
Awa/sun	Aawa/father
Bitaa/pessimism	Biittaa/earth/land
Bora/criticize	Boora/ox
Essa/stop	Eassa/honey

Consonants are letters that we read in a word. They must be combined with vowels to form a word. In the Wolaita language, there are twenty-nine consonant letters. We read these letters in two ways. The ways are fastening and stressing. The consonants will be single for words that are fastening and double for the stressing words. For example, the stressing word “maataaa” (grass) is fastening. There the consonant “t” is single but in the word “maattaa” (milk) “t” should be double. In Wolaita language the use of “f” and “p” are different. That is “p” can be written both at the beginning and at the middle of a word. But "f" is never written at the beginning, it is written at the middle of a word.

2.4. Wolaita Language Sentence Structure

In Wolaita Language a sentence is a set of words that contain:

1. A subject (keettaawaa) (the topic of the sentence)
2. A predicate /Kettaawabaa yootiyagaa/ (what is said about the subject) For example:-

Masppini yiis (Mesfen has come). Masppini = subject yiis = predicate

Wolaita Language sentences are of five different kinds which are given in the following table:

Table 4: Wolaita Language sentence kinds

Maayo/Affirmative	Eg. Naati kifiliyan doosona /Children are in the class.
Phalqiyo/Negative	Maahe maataa meenna. /A tiger doesn't eat grass.
Oysha/Interrogative	Kassi awude yiide? /When did kassa come?
Kittiyo qofa/ Imperative	Keeta pitta. /sweep the room.
Xoqqiso/Exclamatory	Xosso pala! Oh, my God

Wolaita Language sentences can be classified as simple (hada) and compound (walaha) sentences. Simple (hada) sentences have only one verb. For example: (The boy has **come**). In the case of walaha/undduwaallo compound/complex/ sentence, the sentences have more than one peesho (verb). For example: Taani tukkiya essiyode a keettaa pittawusu (When I **prepared** coffee, she **cleaned** the house.)

2.5. Word Classification (qaalaa shaahota)

There are thousands of words in any language. But not all words have the same job. For example, some words express action. In other words, express a thing. Other words join one word with another word. These are the building blocks of the language. When we want to build a sentence, we use different types of words. Each type of word has its job.

To determine the category of the word, linguists use morphological (internal structure of a word), syntactic (structure of the sentence), and semantic (meaning of a word) as clues.

2.6. Wolaita Language Word Classes

Words in Wolaita Language can be divided into two broad categories: closed (gordda) class types and open (doya) class types. Closed (gordda) classes are those that have relatively fixed

members. While open (doya) classes are those continually changed or borrowed from other languages.

According to, Wolaita Language has six-word classes: nouns (sunttaata), adjectives (sunqqota), Verb (peeshuwa), adverbs (peesho zemppo), preposition (somiya), and pronouns (sunttasota). Some of these are divided into other sub-classes. For example, pronoun class is categorized as a personal pronoun (buzo suntaduwa), demonstrative, and interrogative pronoun (besonne oysha sunttaduwa). A detailed description of Wolaita word classes used by [33] is given in the subsections.

2.6.1. Noun (Sunttaa)

Noun (suntta) is a part of speech that we give to name a person/asaa, place/sohuwa, thing/yohuwa, quality/boolimaa, quantity/payduwa, or concepts/geeraa. Wolaita Language nouns, like English nouns, are words used to name or identify a person, place, or thing. For

Example:

- persons/Asata: Asa(Person),Masppina (Mesfen), Wonddima (Wondemu)
- Place/sohuwa: Amarka (America)
- Things/yohota: Keettaa (house), mittaa(wood)

In Wolaita Language noun classes do have four sub-classes [33]. Table 5: shows the sub-class of nouns in Wolaita Language.

Table 5: Sub-classes of noun

No	Sub-classes	Example
1	Buzo/Proper noun	Toophphiya(Ethiopia), Damota (Damota)
2	Kotta/Common noun	Deshsha (Goat), tuussaa(pillar)
3	Shisho/Collective noun	Katta(food) , dozza(Plant)
4	Geema/Abstract	Carkuwaa(wind), eraa(knowledge)

In Wolaita Language few word endings show a word is a noun, For example:

- tettaa:- Cinccattettaa (wisdom), tummatettaa (truth), yelagatettaa (youth)
- yagaa :- maadiyagaa(helper), miyagaa(eater), yiyagaa(comer)
- etta :- zeretta (seed), pittetta (rubish), gaytotetta (connection)

But this is not always true for every word endings of tettaa, yagaa, etta**,** and others. For example:

- Marotetta layitta (merciful) /Adjective = marotetta/merciful
- agga daboyagga (relate) verb= daboyagga/relate

As we can see from the above example, the words are ended with –etta, yagga endings but they represent different word classes.

2.6.2. Verb (Peesho)

Wolaita language has a subject/ keettaawaa, object/kaalliyagaa, verb/peeshuwa (SOV) word order. The verb/peeshoy is a word that describes the subject's action or state within a sentence. For example: Gebire Mikaeli zino giya biis. (Gebremichaeal went to the market yesterday). Wonddimu ba so oosuwa oottiiddi de'ees. (Wondemu is doing his homework).

2.6.3. Adjective (Sunqquwa)

An adjective/sunqqoy is a word that describes, identifies, or quantifies a word by preceding the noun or pronoun that it modifies. For example bootta (white), zo'o (red), karetta(black), qeeraa (small), woggaa (large), etc.

- Bootta godariya (white hayna)
- Wogga keettaa (large house)

In the above sentence, bootta and Wogga are adjectives that describe the nouns godariya and keettaa respectively.

2.6.4. Pronoun (Sunttata)

In Wolaita Language, pronouns/sunttati are small words that take the place of a noun. We can use a pronoun instead of a noun. Pronouns are words like ne (you), i/a(he/she), nu (we), etc. It has three sub-classes.

Table 6:- Sub-class of pronoun/sunttata.

No	Sub-Class	Example
1	Personal pronoun(asa)	Ta/taani, nu/nunni, ne/nenni, intte, I, a ,eti
2	Demonstrative Pronoun(beso)	Neega, iigaa, nuugeeta, taagaa, nu, nuussi
3	Interrogative pronoun(Oyshsha)	Aybee,awugee, ooni, oona, oogee
4	indefinite (Kalkko)	Amara, issuwa, ayibinne, oonee, ubbaa

2.6.5. Preposition (somiya)

In Wolaita Language prepositions are words usually coming in front of, a noun/sunttaa or pronoun/sunttata and express source, destination, location, and relation to another word or element. For example:

- Taappe **simmin** (after me)
- Neeppe **Kase** (before you)
- Alggaappe **garssaara** (under the bed)

2.6.6. Adverb (Penqquwa/Peesho Zemupuwa)

An adverb/peesho zemppoy is a word that tells us more about a verb. It qualifies or modifies a verb, adjective, and other adverbs. In Wolaita Language modifiers of verb or verb phrases usually express time, location, manner, etc. For example:

- Qolchchi zino yiis. (Qolchcha came yesterday)
- Qolchchi eesuwan yiis. (Qolchcha came quickly)

In the first example, the word zino (yesterday) is used as an adverb of time while in the second sentence; the word eesuwan (quickly) is used as an adverb of manner.

2.7. Wolaita Language Tag sets

The previous section gave a broad description of word classes that Wolaita Language's words fall into. In this section, the actual tags used in this thesis are discussed.

As far as the researcher's knowledge is concerned, there is no publicly available tag set for the Wolaita language. To identify and develop a tag set for this thesis, the researcher has conducted interviews and continuous discussions with Wolaita Language/professionals and other instructors at Wolaita Sodo University.

The tag sets that are discussed below are classified as basic classes and sub-classes of the basic classes are noun/sunttaa, verb/peeshuwa, adjective/sunqquwa, pronoun/sunttaduwa, adverb/peesho zemppo, preposition/dabboyuwa is considered. In addition to these, conjunction/gatoy, interjections/garssa gatoy, numerals/paydotinne punctuations/malaatati are also included as basic classes of the Wolaita language.

Noun and its sub-classes/sunttaanne suntta shaahota

In Wolaitato language nouns have different attributes like numbers, genders, and definiteness which can be common nouns (kotta sunttaa), abstract nouns (ayfiyan beettenna sunttaa), collective nouns (shiiqo sunttaa) and proper noun (buzo sunttaa). Due to the tag set complexity problem, we did not include the entire attribute except for proper nouns. In this main class, we identify nouns as a general class and nouns with conjunction, as a sub-class. This class and its sub-classes are explained in the following examples.

- ✓ Nouns that represent the name of a person/asaa, place/sohuwa, thing/yohuwa, organization/eqotaa, and proper nouns, etc. are considered to be proper nouns and tagged by NN. For example: **Arbbaamincce** (Arbaminch), Bassa (Baassa)
- ✓ Nouns affixed with conjunction are considered as noun conjunctions and tagged by NC/SG. For example Baassanne Qolchcha (Bassa and kolcha)
- ✓ Nouns suffixed with prepositions are considered to be noun preposition sub-class and tagged as NPERP. For example: kaamiy**an** (by car)

Verb and its sub-classes/peeshuwaanne peeshuwa shaahota

In this main class, one general class and four sub-classes of a verb are identified. This class

Its sub-classes are explained in the following example.

- ✓ Verbs attached with conjunction categorized under verb conjunction sub-class and tagged by VC. For example Miidi uyiis (ate and drank)
- ✓ Verbs that cannot be classified under the above classification, are categorized under the general tag i.e. verb, and tagged by VV.

Adverb and its sub-classes

Wolaita Language adverbs are words that qualify or modify a verb, adjective, or other adverbs.

In this thesis, we identify one general class and two subclasses. This class and its sub-class are explained using the following example.

- ✓ Adverbs attached with conjunction are tagged by ADVC. For example Naagetissiyogaanne (warning and)
- ✓ Adverbs attached with prepositions are tagged as ADVPREP. For example ayba

ogiyani (in what way)

- ✓ Other forms of adverbs that cannot be classified under the above classifications are tagged as AD.

Adjective and its sub-classes /sunqquwaanne A shaahota

In this main class, we identify one general class and three sub-classes. This class and its sub-classes are explained in the following example.

- ✓ Adjectives attached with conjunction are tagged by JC. For example issippetettanne ikkonome (social and economic)
- ✓ Adjectives attached with Numerals are tagged as JN. For example: ISSI (ONE)
- ✓ All other forms of adjectives that cannot be classified under the above classification are tagged by JJ.

Numerals

Wolaita Language numerals like that of English, Amharic, and Afaan Oromo can be cardinal or ordinal which are tagged as CN and ON respectively. Numerals can be attached with conjunction. If the cardinal is attached with a conjunction, they are tagged with CNC respectively. If the ordinal is attached with the preposition, they are tagged with ONC respectively.

Preposition and its sub-classes

Wolaita Language, prepositions/dabboyoti alone does not convey any meaning unless they are attached to other basic classes like nouns, adjectives, adverbs, etc. In this thesis; we try to identify prepositions as a general class and prepositions attached with conjunction as a subclass.

This class and its sub-class are explained in the following example.

- ✓ Prepositions attached with conjunction are classified under preposition conjunction and tagged as PRC. For example **giddo colbbuwan** (at the middle) or **xeeran** (at the edge)
- ✓ Prepositions other than the one mentioned above are tagged by PR.

Conjunction/gattiyageeta

Conjunctions/Gattiyageeti are words that serve to connect words/qalata, phrase/paca qofa qashota, clauses, or sentence/qofa qashota. In this thesis, conjunctions that can be used as a separate word are tagged as CC. Example: issibaa ootti simmin (after doing something) woykko (or).

Interjections/garssan siyettiyabaa qonccissiyageeta

Interjections/ are words used to express strong feelings or sad emotions. All words that show this type of characteristic are tagged as II. Example: aayyee, waano.

Punctuations /Malaatata

All Wolaita Language punctuation marks such as., ?, !, „ and “ are tagged by MAL./PN. The summarized version of Wolaita Language tag sets which are used to tag untagged Wolaita Language texts are shown in the following table:

Table 7: Tag sets for Wolaita Language languages.

Tag	**Description**
NN	A tag for all types of nouns that are not joined with other categories in sentences.
NC	A tag for all nouns that are not separated from conjunctions.
PP	A tag for all pronouns that are not joined with other categories.
PC	A tag for all pronouns that are not separated from conjunctions.
VV	A tag for all main verbs in sentences.
JJ	A tag for all adjectives that are separated from other categories.
JC	A tag for adjectives that are not separated from conjunction.
JN	A tag for numeral adjectives.
AD	A tag for all types of adverbs in the language.
PR	A tag for all prepositions/postpositions that are separated from other categories.
ON	A tag for ordinary numerals.
CC	A tag for all conjunctions that are separated from other categories.
II	A tag for all introjections in the language.
PN	A tag for all punctuations in the language.
ONC	Ordinal with conjunction
CNC	Cardinal with conjunction
PRC	Preposition with conjunction

CHAPTER THREE:

RESEARCH METHODOLOGY

3.1. Introduction

Part of speech (POS) tagging is the process of allocating the part of speech tag to every word in a sentence for a given language [2]. In many Natural Language Processing presentations such as word intellect disambiguation, information recovery, information handling, analyzing, interrogating, and machine interpretation, POS tagging is reflected as one of the basic obligatory tools [11]. There are many potential distinctions we can draw leading to potentially large tag sets. To do POS tagging, we need to choose a standard set of tags to work with. We could pick very common tag sets such as NN, VV, JJ, PP, and AD.

Wolaita Language POS tagging is a method of assigning a specific Wolaita Language part of speech tag to each word in a Wolaita Language sentence to disambiguate the function of that word in the specific context. In the previous chapter, we discussed the general word categories and tag set for the Wolaita language which is implemented in this chapter. As discussed in the previous chapter in detail, seventeen tags have been identified for the study. For instance, all subtypes of nouns in the language are assigned NN except nouns joined with conjunctions that are assigned NC respectively. So, the implementation of the study is based on these seventeen tags that have been identified as a tag set. Besides the absence of a tag set for the language, the lack of a corpus that represents the language is also another great issue in the study. So the preparation of the corpus is one concern that is discussed in this chapter because the corpus is also one of the basic components of the study. There are three general approaches to dealing with the tagging problem: The rule-based approach, the Statistical approach, and the Hybrid approach. The Rule-based approach consists of developing a rules knowledge base established by linguists to define precisely how and where to assign the various POS tags. The statistical approach consists of building a trainable model and using the previously-tagged corpus to estimate its parameters. Once this is done, the model can be used to determine the tagger of other texts. Generally, successful statistical taggers are mainly based on Hidden Markov Models (HMMs). Finally, the hybrid approach consists of combining a rule-based approach with a statistical one. Recently, most of the POS Taggers have used the latter approach as it gives better results. For this study, we proposed a Part-Of-Speech Tagging method based on a hybrid approach that combines the rule-based approach

with a statistical approach (Hidden Markov Model).

3.2.Corpus Preparation

Corpus, plural corpora, is a collection of text. It can be a flat text i.e. a text with no additional linguistic information or a text whereby each word in the text is attached with linguistic information [34], [6], [35] and [11]. The corpus with additional linguistic information can be called an annotated/tagged corpus. Such linguistic information in the annotated corpus can be part of speech information, sentiment information that specifies the word's word class category and sentiment category respectively. The annotated corpus can be used in many NLP applications part of speech tagger training and testing, parsing, sentiment analysis, etc. In this thesis work, the annotated corpus used is considered to be a text tagged with the corresponding part of speech tags.

The tagged text i.e. the annotated corpus is thought to represent all the domains of the language. The domains can be the text of news category, fiction category, editorial category, scientific category, etc. A corpus with all possible categories is called a balanced corpus. Though it is difficult to prepare a balanced corpus, it is an important element in most natural language processing applications in general and part of speech tagging in particular [35].

Developing a balanced corpus takes time, effort/skills of language experts, and money as it needs data to be collected from different domains. Due to these constraints, a balanced corpus is developed for this thesis work rather than three or more categories, the news category; Bible, Textbooks, and others are selected as texts are available and can be collected from different sources. The essence of developing a balanced corpus is to increase the performance of the tagger when it tags any text taken from any category which implies directly that a balanced corpus contains as many words as possible from different categories in their appropriate sense. However, a category-specific corpus contains words that are mostly used in that category, and if a text from other categories is to be tagged is given to the tagger trained on this corpus. But if the text taken is from that category, the performance is assumed to perform as expected [35].

There are three main stages for incremental corpus preparation these steps can be summarized as manual, automatic, and correction stage. In the manual stage, the text collected from different sources is passed to the annotators for manual annotation and the output of the manual annotation is used to train the tagger. After this, in the automatic stage, the tagger, which

is trained on the manually annotated text, tags new raw text. In the correction stage, the output of the trained tagger is passed to the annotator for manual correction. The output that is obtained from the correction stage is added to the training set to contain the approved text which is used for training the tagger. Starting from the automatic stage, the process is repeated until the desired corpus size is achieved. Figure 1 show the stages involved in the incremental corpus preparation approach for word class-tagged corpus preparation.

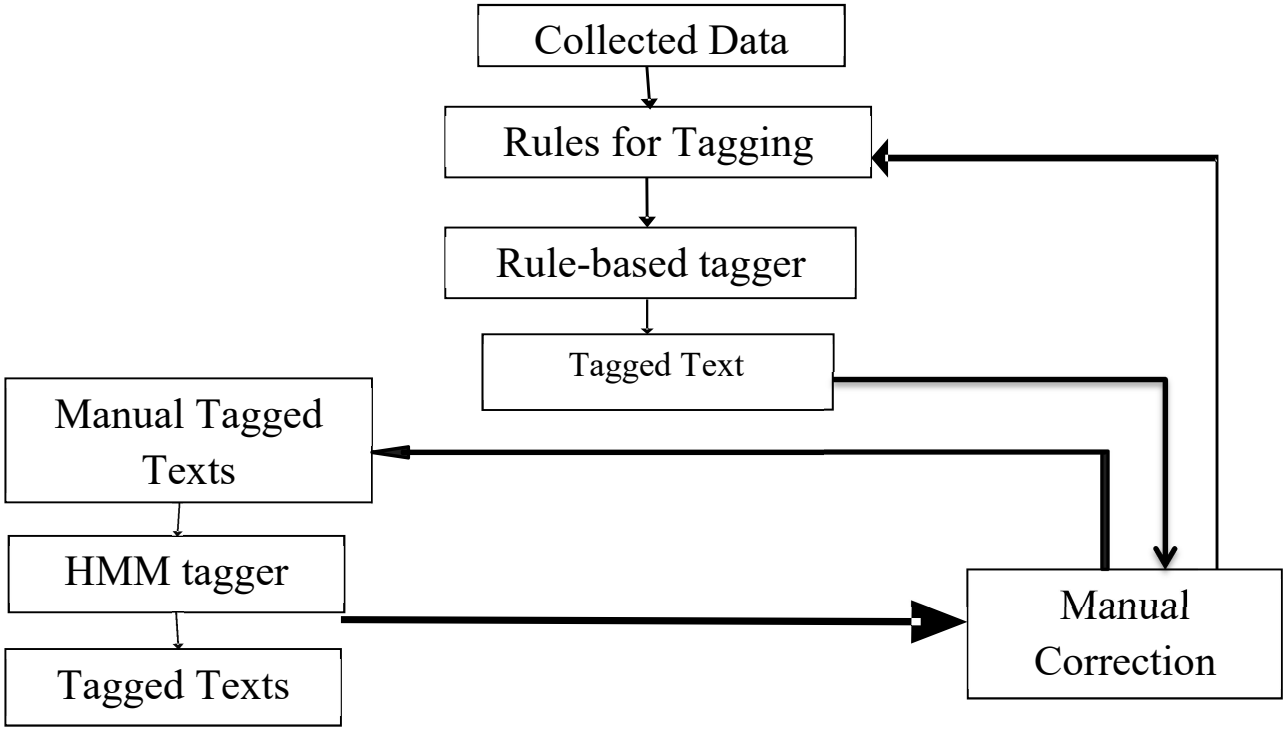

Figure 1: Steps involved in the incremental corpus preparation process.

3.3. System Architecture

Hybrid Approach: Consists in combining Rule-Based method with a Statistical method used to assign the best tag for each of the words of input text. One of the most recent works for POS tagging is done by Jabbari and Allison [36]. Their approach is transformation-based and has previously been used in English by Brill and Hepple [17], [37]. The construction of this tagger contains a trained learner machine which includes approximated rules.

However, this rule-based method [38] presents some problems: it may misclassify and not analyze some words. This paper proposes a Wolaita Language Part-Of-Speech Tagging method which combines the Rule-Based approach with the HMM technique. The latter is superior to other models in terms of training time and is suitable for applications dealing with large amounts of text. In the

following two sections, both Rule-based Tagging and HMM-based Tagging methods will be presented and detailed.

Rule-based and stochastic-based approaches are the two widely used approaches to POS tagging. The rule-based approaches use contextual information to constrain the possible part of speech tags or to assign a part of a speech tag to a word. The statistical (stochastic) approaches select the most likely interpretation based on the estimation of statistics from unambiguously tagged text.

3.3.1. Designing a rule-based tagging method

The Rule-Based tagging method allows labeling the words in a wolaita language to their tags. It is constituted of three main phases: the lexicon analyzer, the morphological analyzer, and the syntax analyzer. Figure 2 shows the architecture of this system.

Lexicon Analyzer: In this step, a lexicon of stop lists in the Wolaita language is defined. This lexicon includes prepositions, adverbs, conjunctions, interrogative particles, exceptions, and interjections.

All the words have to pass this phase. If the word is found in the lexicon, it is considered as tagged. Else, it passes to the next step.

Morphological Analyzer: Each word that has not been tagged in the previous phase will immigrate to this phase. A set of the affixes of each word are extracted. An affix may be a prefix, suffix, or infix. After that, these affixes and the relations between them are used in a set of rules to tag the word into its class. Note that this phase is the core of the system since it distinguishes the major percentage of untagged words into nouns or verbs.

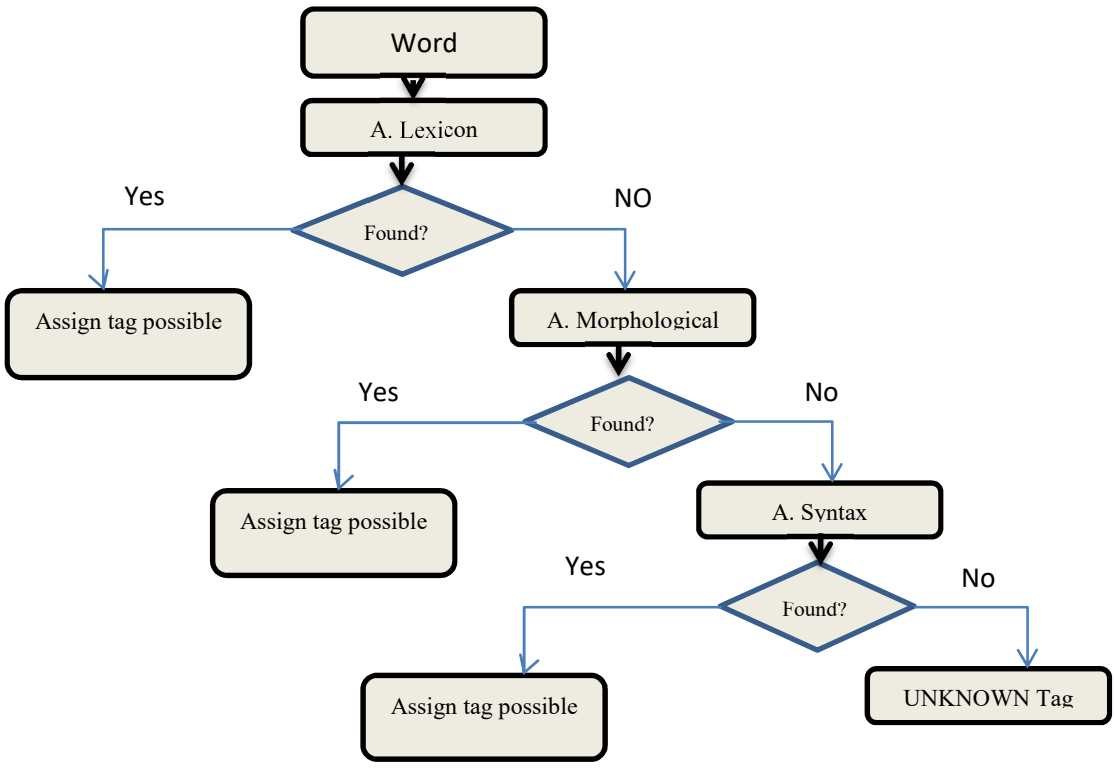

Figure2. The architecture of the Rule-Based Wolaita Language POS Tagger

Syntax Analyzer: This phase can help in tagging the words that the previous two phases failed to tag. It consists of two rules: sentence context and reverse parsing. The sentence context rule is based on the relation between the untagged words and their adjacent. Wolaita language has some types of relations between adjacent words. For example, prepositions are more times followed by nouns. These relations may allow tagging the words into their corresponding classes. The reverse parsing rule is based on Wolaita context-free grammar. The authors propose a set of rules that are used frequently in the Wolaita language.

3.3.2. Designing Hidden Markov Model (HMM) Tagger

This section covers the use of a Hidden Markov Model (HMM) to do part-of-speech tagging can be seen as a special case of Bayesian inference [39]. It can be formalized as follows: for a given sequence of words, what is the best sequence of tags that corresponds to this sequence of words? If we represent an entered text (sequence of morphological units in our case) by $\mathbf{W} = \mathbf{(W_i)_{1sisn}}$ and a sequence of tags from the lexicon by $\mathbf{T=(t_i)_{1sisn}}$, we have to compute:

$$\mathbf{Max[P(T\backslash W)]} \quad \text{.....(1)}$$

By using the Bayesian rule and then eliminating the constant part, P(T) the equation can be transformed to this new one:

$$\max_{T} [P(T\backslash W)*P(T)] \ldots\ldots (2)$$

Where P(T) represents the probability of the tag sequence (tag transition probabilities) and can be computed using an N-gram model, as follows:

$$\boldsymbol{P(T = t_1 t_2 \ldots\ldots t_n) = \prod_{i=1}^{n} P(t_i \backslash t_{i-n} \ldots t_{i-2} t_{i-1}} \ldots\ldots \textbf{(3)}$$

A tagged training corpus is used to compute $P(t_i\backslash t_{i-n} \ldots\ldots t_{i-2}t_{i-1})$, by calculating frequencies of N-gram as follows:

$$\mathbf{P(t_i\backslash t_{i-n}\ldots\ldots t_{i-2}t_{i-1}) = (p(t_{i-n}\ldots\ldots t_{i-2}t_{i-1})|P(t_{i-n}\ldots t_{i-2}t_{i-1}))}\ldots\ldots\textbf{(4)}$$

However, it can happen that some trigrams (or bigrams) will never appear in the training set; so, to avoid assigning null probabilities to unseen trigrams (bigrams), we used a deleted interpolation developed by [20]:

$$\boldsymbol{\lambda_1 P(t_i \backslash t_{i-n} \ldots . t_{i-n-1}) + \cdots + \lambda_{n-2} P(t_i \backslash t_{i-2} t_{i-1}) + \lambda_{n-1} P(t_i \backslash t_{i-1}) + \lambda_n P(t_i)} \ldots\ldots \textbf{(5)}$$

Where $\boldsymbol{\lambda_1 + \lambda_2 + \cdots + \lambda_n = 1}$

Then, for calculating the likelihood of the word sequence given tag, the probability of a word appearing is generally supposed to be dependent only on its part-of-speech tag. So, it can be written as follows:

$$P(W\backslash T) = \prod_{i=1}^{n} P(w_i \backslash t_i) \ldots\ldots (6)$$

In addition, a tagged training set has to be used for computing these probabilities, as follows:

$$P(P(w_i\backslash t_i) = \frac{f(w_i,t_i)}{f(t_i)} \ldots\ldots (7)$$

Where $f(w_i,\ t_i)$ and $f(T_i)$(ti) represent respectively how many times is tagged and the frequency of the tag t_i itself.

Tag sequence probabilities and word likelihoods represent the HMM model parameters: transition probabilities and emission (observation) probabilities. Once these parameters are set, the HMM

model can be used to find the best sequence of given a sequence of input words. The Viterbi algorithm can be used to perform this task.

The HMM strives to find the optimal sequence of part-of-speech tags for a sequence of words in an input sentence using the Viterbi algorithm. The tagger gets the lexical probability and contextual probability from the training corpus. To train the HMM tagger, we use a supervised learning mechanism, where a pre-tagged corpus is a prerequisite. The process follows three steps. In the first step, we provide a tagged corpus as training data. The tagged data passed through a pre-processing module (sentence and word tokenize) to be tokenized both at sentence and word level respectively. In the second step, the output of the pre-processing component is passed to the statistical analyzer module that computes both lexical and contextual probabilities and stores them in the database. Finally, we provide a list of tags and store them in the database. After the tagger is trained, it is used for annotating untagged sentences which can in turn be evaluated against the manually tagged data (reference data) of the input untagged sentence.

The graphical representation of the tagging and evaluation of the tagger is shown in the figure below.

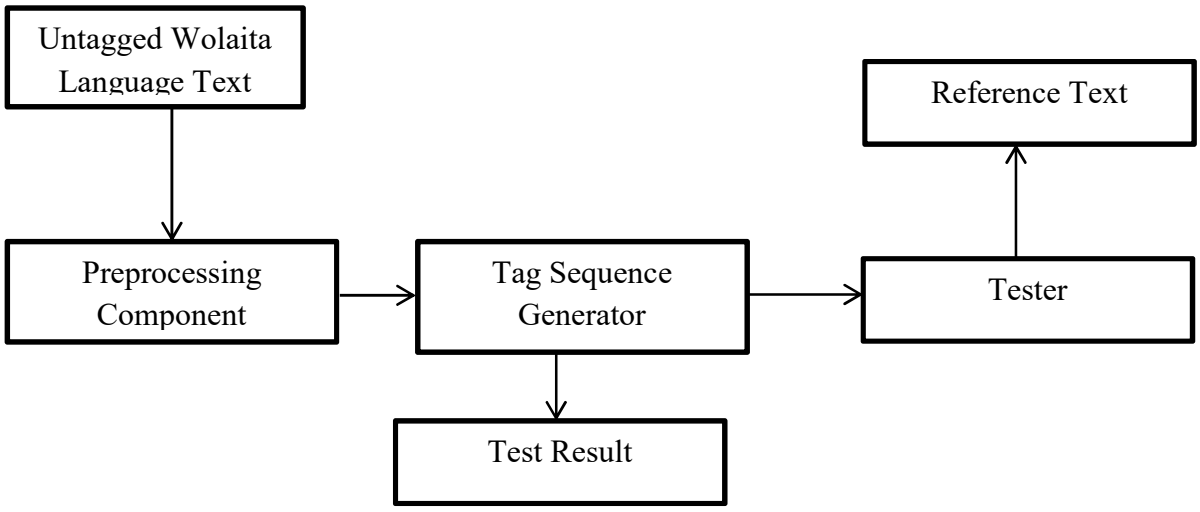

Figure 3: The HMM tagger trainer model.

The untagged text is given to the sentence splitter module and Tokenizer preprocessing components to make it ready for tagging by the Tag Sequence Generator. Afterward, the Tag Sequence Generator selects an optimal part of the speech tag sequence for the given word sequences and gives the tagged word sequences as an output. The tagged text is given to the tester component for comparison against a manually tagged (so-called reference text) and this

component gives accuracy of the tagging by counting the number of correctly tagged words.

The lexical and contextual probabilities that are stored within the database provide information about the probability of each tag given the word and information about the context of the word respectively. Based on the stored information, the initial-state tagger assigns an optimal part-of-speech tag for a given word sequence using a table-driven method called the Viterbi algorithm and provides the tagged text as an output.

3.4. **Our method for wolaita language**

The proposed POS Tagger for the Wolaita Language is based on a hybrid approach; it combines the Rule-based method with an HMM model (see Figure 4). As we have mentioned, the Rule-based method is composed of three steps: lexicon analyzer, morphological analyzer, and syntax analyzer. Almost all words are recognized by the rule-based method. However, some terms are not analyzed or misclassified. These terms (the rest failed terms) will be analyzed using the HMM model.

The states of this model correspond to part-of-speech tags and the observations correspond to words (see Figure 5). The basic idea of our HMM model which adopts supervised learning is to assign the most probable tag to the word of an input sentence. Two major steps are required: the training step and the test step.

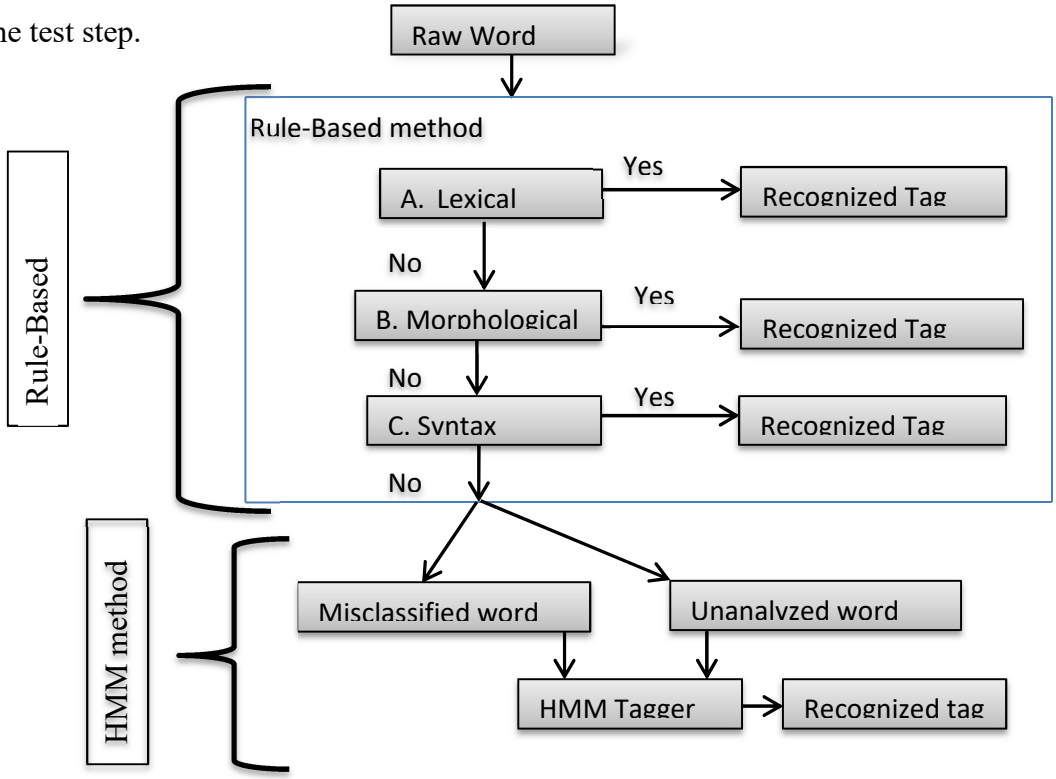

Figure 4. Flowchart of the Proposed Wolaita Language POS Tagger

The Training Step is based on supervised learning. It allows us to learn the parameters of the HMM model using the corpus by estimating the transition and emission probabilities. First, for each iteration (concerning one term) we compute the emission probability for each tag i.e. p (word |tagi) (see equation 7). Second, for each iteration (concerning one tag) we calculate the transition probabilities which represent the relation between the tag and the previous tag i.e. p (tag| previous tag) (see equation 7) for the Hidden Markov Model. The results of this step are two matrices: the matrix of transition probabilities (Tag/ Tag) and the matrix of emission probabilities (word/Tag).

The Testing Step aims to assign the best probable word's tag for which the term has been misclassified or unanalyzed during the rule-based process. First, we give all possible tags for this word. Then, we compute the probabilities of each tag of this word by using the transition probabilities and emission probabilities. Finally, the Viterbi algorithm is used to calculate the best probable path (best tag sequence) for a given word in a sequence (sentence).

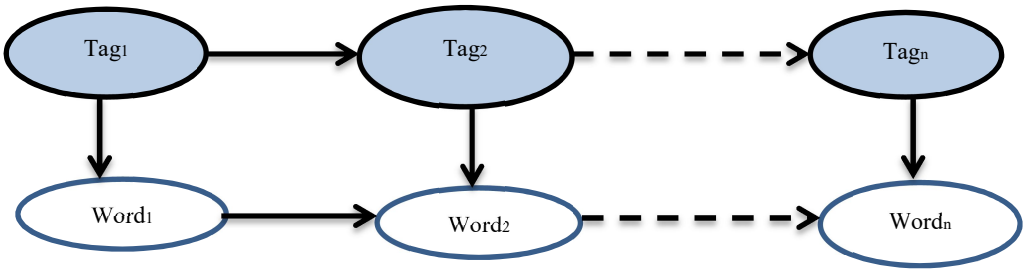

Figure 5: The Architecture of HMM Tagger

3.5.Part-of-speech tagger Tool

Natural Language Toolkit (NLTK) and Python are used in the entire implementation of the Tagger. The rationale behind the choice of these two tools is they are suitable for processing different NLP tasks [40]. NLTK is an open-source tool that contains open-source Python modules, linguistic data, and documentation for research and development in natural language processing [40], [41]. It supports many NLP tasks such as tokenizer, stemmer, part of speech tagger, and classifier with distributions for Windows, Mac, and Linux for some languages such as English and German. Moreover, it contains different corpora for these languages with their respective corpus reader module. Python is an easy-to-learn but powerful programming language especially for text processing in NLP applications [4], [34] and [9]. It has efficient high-level data

structures and a simple but effective approach to object-oriented programming [41]. Its elegant syntax and dynamic typing, together with its interpreted nature make it an ideal language for scripting and rapid application development in many areas particularly in natural language processing on most platforms [40]. In this study, preprocessing algorithms including sentence splitter word tokenizer are used.

CHAPTER FOUR: RESULTS AND DISCUSSIONS

4. Results

4.1.Performance Analysis of the Tagger

The researcher used percentage split evaluation method in experiment. According to this percentage split evaluation method, the entire corpus is split into two sets: Training set and testing set. To test the performance of the tagger the following experiments are carried out. The first experiment is done to validate the tagger how much it performs on a portion of the training set. Out of the total data set 10% is used for testing and 90% is used for training.

The second performance analysis uses tenfold cross-validation. Tenfold cross-validation divides a given corpus into ten folds. And nine folds are used for training and the tenth fold is used for testing. It provides an unbiased estimate of the value of prediction error and is preferred for small sample corpus [10].

4.1.1. Performance Analysis with Rule based and HMM model tested By Train-set

The table below indicates the performance of the tagger which is tested on 10% of the total training set. The purpose of using this validation method is to evaluate the human errors while tagging the sample corpus manually besides validating the tagger. Accordingly, after correcting human and spelling errors, the performance of the tagger shows 66069 words (95.77%) correctly tagged words for Rule Based and the performance of the tagger shows 2319 words (99.2%) correctly tagged words for HMM model.

Table 8: Validating the Tagger with 90% Train_ Set.

Total Tagged words in Training	Correctly tagged Word	Incorrectly tagged Word	Accuracy in Percent	
69547	66069	3476	95.77%	By Rule Based
2367	2319	48	98.2%	By HMM

Incorrectly tagged words are analyzed through manually tagged corpus. Most of them are tagged incorrectly from nouns to adjectives and vice versa. As discussed in Chapter Three, categories of words in sentences characterize their position. In particular, some noun words take the position of adjectives and vice versa in different contexts of sentences. The other errors are due to human errors while preparing the manually tagged corpus. Such kinds of problems are handled

by reexamining the corpus and correcting wrongly tagged words according to their positional function in sentences. The other problem that causes a wrongly tagged assignment is spelling errors. In the same words, there are different spellings in the training set and test set. These problems can be solved by developing a standardized corpus for the language.

4.1.2. Performance Analysis with Rule based and HMM model tested by Test Set

In this performance analysis, the tagger is repeatedly trained and tested following tenfold cross-validation as shown in the following Table.

Table 9: The Accuracy of the Rule-based Model Tested by Test Sets

Total Test sets	Correctly tagged words	Incorrectly tagged words	Accuracy in percent	No of Test Set
7728	6955	773	90.5%	Rule based
Average Accuracy			90.5%	

Table 10: The Accuracy of the HMM Model Tested by Test Sets.

Total Test sets	Correctly tagged words	Incorrectly tagged words	Accuracy in percent	No of Test Set
283	277	6	98.4%	
		Average Accuracy	98.4%	

The model of the tagger is tested with a corpus of up to 7728 Wolaita Language words on average in each test set and that is trained on the training set of more than 6955 words in Rule based model and the other of HMM is more than 283 Wolaita Language words on average in each test set and that is trained on the training set of more than 2367 words, and the result of each test is compared with a copy of the test set that is by using rules. As a result, the results of the experiments for both Rule bases and HMM algorithms show an accuracy of 93.13% and 98.3% correctly tagged words on average respectively.

With this corpus, the distributions of accuracy performance in both models are not as far from each other. If the corpus is standardized, this variation will reduce since a standardized corpus consists relatively complete representative of words for the language, and a fair distribution of words in the training set and test are observed.

4.2. **Discussion of Results**

The main purpose of this study is to design and develop a POS tagger for the Wolaita-language. For this thesis, a corpus with a total of 79,925 words is collected from the same of domains. There is several standard tag sets used in corpora and in POS tagging experiments.

For this thesis, 17 part-of-speech tags are identified as a tag set for annotating a raw text. The tag set indicates word classes rather than gender, number, tenses, etc.

The tagged corpus is divided into training and testing sets. The training set comprises 90% of the entire corpus the remaining 10% corpus is used as a testing set.

NLTK version 3.8.1 and Python version 3.11 are used in the implementation and experiment of the Wolaita Language part-of-speech tagger.

CHAPTER FIVE: CONCLUSIONS AND FUTURE WORKS

5.1. Conclusion

Part-of-speech (POS) tagging for Wolaita language is the process of assigning the appropriate part of speech or lexical, Morphological, and syntax category to each word in Wolaita language sentence. Part-of-speech tagging is an important part of Natural Language Processing (NLP) and is useful for most NLP applications. It is a research area in the field of natural language processing for different languages. Several approaches have been proposed to annotate words automatically with their part-of-speech tags. For this study, we propose a Part-Of-Speech Tagging method based on a hybrid approach that combines the rule-based approach with a statistical approach (Hidden Markov Model). The study is used as groundwork for parts of tagger development for the Wolaita language. Since there are no corpus and tag sets for natural language processing of the language, sample corpus and tag set are designed and developed for the study.

Corpora are important for many types of linguistic research including part-of-speech tagging. For this thesis, a corpus with a total of 79,925 words is collected from the same of domains.

There is several standard tag sets used in corpora and in POS tagging experiments. For this thesis, 17 part-of-speech tags are identified as a tag set for annotating a raw text. The tag set indicates word classes rather than gender, number, tenses, etc.

The tagged corpus is divided into training and testing sets. The training set comprises 90% of the entire corpus the remaining corpus is used as a testing set.

NLTK version 3.8.1 and Python version 3.11 are used in the implementation and experiment of the Wolaita Language part-of-speech tagger.

5.2. FUTURE WORKS

There are lots of researches interests in the areas of natural language processing that can be done for local languages. Among these, part-of-speech tagging is a useful form of linguistic analysis. It serves as a pre-processing component for many higher-level NLP applications such as spelling checkers, grammar checkers, question answering, etc. Therefore, the results presented herein can function as a first benchmark for future research on part-of-speech tagging of the Wolaita language. In a future work, we would like to suggest the following key points:

- The accuracy and effective processing of natural language processing that needs annotated data sets depends on a standardized and sufficient amount of corpus. The lack of a standardized annotated corpus is one of the limitations of the language. So, the responsible agents should take it into consideration to prepare it.
- Extending this work by training the system in a large corpus and using a large tag set that can identify different features such as gender, number, tense, etc.
- Other approaches are different from the Hidden Markov Model which can be applied to similar problems. So, experts and researchers can develop the same title for the language in rule-based and neural networks to compare the result with the performance of this study.
- Currently, many part-of-speech taggers have been done in combination with both the hidden Markov Model and Rule-based approach to get more accurate results and to handle rare words and unknown words. So, to make full-fledged the tagger, experts and researchers can do it in combination with the Hidden Markov Model and Rule-based tagging approach.

REFERENCE

1. *Name nouns in Wolaitta. I.* **Hayward, Richard.** In Omotic Language Studies ed.: School of Oriental and African Studies., London : s.n., 1990.

2. *"Tamil POS Tagging using Linear Programming", .* **Dhanalakshmi V, Anand Kumar1, Shivapratap G, Soman KP and Rajendran S,.** International Journal of Recent Trends in Engineering,, May 2009.

3. *In Encyclopedia of Library and Information Science, 2nd Ed. , Inc., .* **Decker, Truscott John.Marcel.** "The effect of error correction on learners "ability to write accurately" Journal of Second Language Writing,, 2003.

4. **Baeza-Yates.** Modern information retrieval. (1999). Addison-Wesley Longman.

5. **Yair Halevi.** "Part of Speech Tagging",. April 2006. Seminar in Natural Language Processing and Computational Linguistics (Prof. Nachum Dershowitz),.

6. *String analysis of the language structure. .* **Z., Harris.** Mutton and Co., The Hauge., 1962.

7. *Part of Speech Tagger for Turkish, .* **Levent Altunyurt & Zihni Orhan.** Master's Thesis, Computer Engineering, Bo_azici University., (2006).

8. **Asres., Solomon.** *"Automatic Amharic Part-of-Speech Tagging Using Hybrid Approach (Neural Network and Rule-Based)".* Addis Ababa, : s.n., 2008.

9. *"Part-of-Speech Tagging for Afaan Oromo Language".* **Meshesha., Getachew Mamo and Million.** In: Proceeding International Journal of Advanced Computer Science and applications, special issue on Artificial Intelligence, , 2011, Vols. Vol.1, No. 3, USA, pp.1-5, .

10. *"Part-of-speech tagger for Afaan Oromo language using Transformation Error Driven Learning (TEL) approach".* **Hussen., Mohammed.** Master's Thesis, Addis Ababa University, Addis Ababa, February 2010.

11. **Gebregzabiher., Teklay.** "Part-of-Speech Tagger for Tigrigna Language" . .Addis Ababa, : s.n., November 2010. Vols. Master's Thesis, Addis Ababa University, .

12. *Automatic grammatical taggin of English. .* **Greene B. B. and Rubin G. M.** Technical Report, Department of Linguistics, Brown University., 1971.

13. **Zelalem Mekuria.** Design and development of part-of-speech tagger for Kafinanoo language, . (2013). Master's thesis, Addis Ababa University.

14. *PART OF SPEECH TAGGING FOR AMHARIC.* **Gebre, Binyam Gebrekidan.** Research Institute in Information and Language Processing,

15. *Part of Speech Tagging for Wolaita Language using Transformation.* **Shirko, Birhanesh Fikre.** 2020.

16. **Brill, Eric.** A simple rule-based part of speech tagger. . 1992. In Proceedings of the Third ACL Applied NLP, Trento, Italy, pages 152-155.

17. *Transformation Based Error-Driven Learning and Natural Language processing: case study in Part-of speech Tagging.* . **Brill, Eric.** The Johns Hopkins University, : s.n., (1995). Vol 21,No.4,pp. 543-565.

18. *A computational approach to grammatical coding of English words.* . **R, Klein S. and Simmons.** Journal of the Association for Computing Machinery,, 1963. .

19. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text.* . **Karlsson F., Voutilainen A., Heikkila J. and Anttila.** Mouton de Gruyter, Berlin., 1995.

20. **Nugues., Pierre M.** An Introduction to Language Processing with Perl and Prolog springer. (2006), Springer-Verlag Berlin Heidelberg, Germany.

21. *Natural Language Understanding* . **James, Allen.** Rewood City, Canada : The Benjamin/Comming Publishing Company, 1995.

22. **Getachew, Mamo.** *Part Of Speech Tagging for Afaan Oromo Language* . Master's Thesis, Addis Ababa University : s.n., (2009).

23. **John, Hall.** *A Probabilistic Part of speech Tagger with Suffix Probabilities.* . Masters Thesis, School of Mathimathics System Engineering, Vaxjo University , Smaland, : s.n., Appril (2003).

24. *Comparission of Different POS Tagging Techniques (n-grams, HMM and Brill's Tagger) for Bangla,.* **Muhammed Hasan Fahim, Naushad UzZaman, Mumit Khan,.** s.l. : International Confrance on system, Computing Sciences and Software Engineering (SCS206) , (2006).

25. *Speech and Language Processing, An introduction to natural Language Processing, Computational Linguistics, and speech recognition.* **D. Jurafky and J. H. Martin.** (2006). 2nd Ed. New Jersey, Prentice Hall.

26. **Dandapat, Sandipan,.** *Part-of-speech tagging for Bengali,.* IIT Kharappur : s.n., (2009).

27. *The effect of error correction on learners "ability to write accurately".* **Liddy, E. D.,.** s.l. : Natural Language Processing. In Encyclopedia of Library and Information Science, 2nd Ed. Marcel Decker, Inc.,Truscott John., December 2007.

28. **Simon, Haykin.** *Neural Networks: A comprehensive Foundation. Macmillan, Ontario,.* (1994).

29. **MacKinlay, Andrew.** *The Effects of Part-of-Speech Tagsets on Tagger Performance.* Undergraduate Thesis, University of Melbourne, : s.n., (2005).

30. *"Hidden Markov Model with Rule Based Approach for Part of Speech Tagging of Myanmar Language".* **Zin., Khine.** In: Proceeding of the 3rd international conference on communications and information technology, Florida, : s.n., December (2009). pp. 123-128,.

31. **Ethnologue:, Wolaytta entry in.** *www.ethnologue.com/language/wal/visited on April 1.* (2017).

32. **Adams, Bruce A.** *Name nouns in Wolaitta. In Omotic Language Studies ed.* . 406-412. London: School of Oriental and African Studies. : by Richard Hayward, (1990).

33. *WOLAITA-AMHARIC-ENGLISH GRAMMAR, .* **Isayas Elias.** Addis Ababa, : s.n., (2015).

34. *Implementing an efficient Part-of -speech tagger, .* **V.Kann, Carlberger and.** New York : John Wily & Sons, Inc. , (1999).

35. *Natural Language processing with Python : Analyzing Text with Natural Language Toolkit.* **Steven Bird, Ewan Klein, and Edward Loper.** (2009). 1st Ed.

36. *"Persiant Part-Of-Speech Tagging : .* **B., S.Jabbari &.** USA : In The Proceedings of workshop on computational Approaches to Arabic Script-Based Languages, (2007).

37. *"Indipendence and Commitment: Assumptions for Rapid Training and Rule based Part of speech Taggers".* **M.Hepple.** Hong Kong : In Preeceedings of 38th Annual meeting of Association for computational Linguistics, (2000).

38. *A Rule Based Approaches for tagging non vocalized Arebic words. .* **S.Abu-Al-Rub, Al-Taani &.** (2009).

39. *TnT: A Statistical Part of Speech Tagger, .* **T.Brants.** USA : Association for Computational Linguistics Morristown, (2000).

40. *NLTK: The natural Language toolkit . .* **Bird, S.,.** s.l. : In Proceeding of 3rd COLING/ACL on interactive peresentation session , (2006).

41. *The Natural Language Toolkit ://WWW.NLTK.ORG/.* Visited on April 15,2017.

42. *Power distribution network reconfiguration for loss minimization using a new graph theory based genetic algorithm.* **Raut, U. and Mishra, S.** Calcutta : s.n., 2018. 2018 IEEE Calcutta Conf.

Appendix I

Wolaita language sentences for this thesis used are:-

geeshsha maxaafaa taariketa tamaariyo maxaafaa

ha maxaafay bayzettiyaaba gidenna . alame yuushuwan geeshsha maxaafaa tamaarissanau giigissiyoobaappe issuwaa gidishin , he oosoy asay ba dosan immiyo miish shan oosettees .

geeshsha maxaafaa taariketa tamaariyo maxaafaa

hagee tumu taariketa oyqqida maxaafa . taariketi ekettidoy alamiyan de''iya ayba maxaafatuppekka nashshettida , geeshsha maxaafaappe. ha taariketi xoossay medhdhiyoogaa doommoosappe hanno gakkanaashin alamiyan hanettidabata yootoosona . eti ubba , xoossay wodeppe oottanaayyo qaalaa gelidobatakka yootoosona . ha maxaafay , geeshsha maxaafan de''iya kumetta qofay aybakko qonccissees . etabaymne eti oottidobay geeshsha maxaafan xaafettido asatubaa yootees . i , gannate biittan asay merinau de anaadan xoossay giigissido gita hidootaabaakka erissees . ha maxaafan 116n taariketi de''oosona . he taariketi hosppunan shaahettidosona . shaahotu ubbaayyo doomettan de''iya issi sinttay , he shaahuwaa giddon de''iyabaa qanttan yootees . taariketi xaafettidoy polettido wodiyaa maaraadaana . hegee, issi hanettidabay hara taarikiyaara gatti xeelliyo wode, aude polettidaakko eranau nena maaddees . taariketi wayssenna qaalan xaafettidosona . yelagatoo , intte giddoppe daroti ha taariketa intte huuphe nabbabana danddayeeta . yelidaageetoo , ha taariketa intte guutta naatussi intte zaari zaari nabbabiyo wode, eti ufayttiyoogaa akeekana danddayeeta . ha maxaafay yelagata gidin ,

Appendix II

Python codes for Rule based Pos Tagging For Wolaita language

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on
win32

Type "help", "copyright", "credits" or "license()" for more information.

import nltk

import numpy as np

import pandas as pd

import random

from sklearn.model_selection import train_test_split

from nltk.tokenize import PunktSentenceTokenizer

from nltk.corpus import webtext

from nltk.tokenize import WhitespaceTokenizer

text11 = webtext.raw('C://Users/user/Documents/regexpp.txt')

tk = WhitespaceTokenizer()

geek=tk.tokenize(text11)

sents_2 = tk.tokenize(text11)

patterns1 = [

    (r'.*ona$', 'VV'),              # verb

    (r'.*enna$', 'VV'),               # verb

    (r'.*ana$', 'VV'),              # verb

    (r'.*ena$', 'VV'),              # verb

    (r'.*ees$', 'VV'),              # verb

    (r'.*oos$', 'VV'),              # verb

    (r'.*iis$', 'VV'),              # verb

    (r'.*aasu$', 'VV'),               # verb
```

```
(r'.*aas$', 'VV'),              # verb

(r'.*ana$', 'VV'),              # verb

(r'.*aana$', 'VV'),             # verb

(r'.*aba$', 'VV'),              # verb

(r'.*ido$', 'VV'),              # verb

(r'.*tetta$', 'NN'),            # nouns

(r'.*yagaa$', 'NN'),            # nouns

(r'.*etta$', 'NN'),             # nouns

(r'.*nne$', 'NC'),              # nouns with conjuction

(r'.*an$', 'NC'),               # nouns with conjuction

(r'.*ppe$', 'NC'),              # nouns with conjuction

(r'.*ra$', 'NC'),               # nouns with conjuction

(r'.*kko$', 'NC'),              # nouns with conjuction

(r'.*kko$', 'NC'),              # nouns with conjuction

 (r'[a|i|o|ta|eti|nu|eta|tana|nuna|nuni|inte|ha|he|hegaa|hagaa|hagan|hagaara|hagaassi|hegaara|hegaas
     si|awugee|awan|awaara|awa|awugaa|aybee|aybissi|aybawu|ooni|oonaara|oossi]', 'PP'),   #
     pronouns

(r'[tanara|taappe|taassi|tanaani]', 'PC'),   # pronouns with conjuction

(r'[etaara|etappe|etani|etassi]', 'PC'),   # pronouns with conjuction

(r'[nunaara|nuuppe|nuuni|nuussi]', 'PC'),  # pronouns with conjuction

 (r'[nubaappe|ibaappe|etabaappe|intebaappe]', 'PC'),  # pronouns with conjuction

(r'[intenaara|inteppe|intenaani|intessi]', 'PC'),  # pronouns with conjuction

(r'[bootta|karetta|zo"o|guutta|qiiba|adussa|qantta|wogga|qiiba|qeera|ooratta|gala|mal"o|camo]', 'JJ'),
   # Adjective

(r'[boottaappe|karettaappe|zo"uwappe|booxida|zo"ida|karexxida|mal"ida|gal"ida|qaammida|oorax
     xida|qiibida|cammida|camuwappe|adussappe|qantaape|woggaappe|qiibaappe|qeeraappe|boottaa
     ssi|karetaassi|adussaassi|qantaassi|woggaassi|qiibaassi|qeeraassi|oorataassi|galassi|maluwassi|c
     amuwassi|issipetettaanne]', 'JC'),# Adjective with conjuctive
```

```
(r'[boottaara|karettaara|zo"uwara|adussaara|qantaara|woggaara|qiibaara|qeerara|oorattaara|gal"aara|mal"uwaara|camuwara]', 'JC'),# Adjective with conjuctive

 (r'[simmin|kase|garssan|bollan|xeeran|matan|haahuwan|heeran|lanqen|guyyen|sinttan|giddon|he|ha]', 'PR'),   # preposition

(r'[simmada|kasetada|garssaara|bollaara|xeeraara|mataara|haahuwaara|heeraara|lanqeera|guyyeera|sinttaara|giddoora|hegan|awugan]', 'PRC'),   # preposition with conjuction

 (r'^-?[0-9]+(.[0-9]+)?$', 'JN'), # Adjective with Numerals

(r'[issi|naa"a|heezza|oydda|ichasha|usuppuna|laappuna|hosppina|uddupuna|tamma|isiino|laatama|hastama|oytama|ishatama|usuppun|laappun|hosppun|uddupun|xeeta|sha"a|]', 'ON'),   # pronouns with conjuction

(r'[issinne|naaanne|naaappe|heezzaappe|oyddappe|ichashappe|usuppunaappe|laappunaappe|hosppinaappe|uddupunaappe|tammaappe|laatamaappe|hastamaappe|oytamaappe|ishatamaappe|\xeetaappe|sh"aappe]', 'ONC'),   # pronouns with conjuction

 (r'[issuwan|naaan|heezzaan|oyddan|ichashan|usuppunan|laappunan|hosppinan|uddupunan|tamman]', 'CNC'),   # pronouns with conjuction

 (r'[nne|shun|gedoppe|attin|qassi|gishshawu]', 'CC'),   # Adverb

  (r'[eesuwan|matan|hachi|zino|wonto|ha"i|wonta|maallado|guuran|guurada]', 'AD'),   # Adverb

(r'[aayyee|waanoo|pala|haako|poora|ayeana|ayssi|aybeesha|hayyana|taassa|hayya|hashu|hashshukkonne]', 'II'),   # Interjection

 (r'.[^(a-z0-9)]', 'PN'),   # Punctuation or symbols

  (r'.*','NN')               # nouns

  ]

>>> regexp_tagger=nltk=nltk.RegexpTagger(patterns1)

>>> sents_2 = tk.tokenize(text11)

>>> regexp_tagger.tag(sents_2)

[('geeshsha', 'PP'), ('maxaafaa', 'JJ'), ('taariketa', 'PP'), ('tamaariyo', 'PP'), ('maxaafaa', 'JJ'), ('ha', 'PP'), ('maxaafay', 'JJ'), ('bayzettiyaaba', 'VV'), ('gidenna', 'VV'), ('.', 'AD'), ('alame', 'PP'), ('yuushuwan', 'NC'), ('Geeshsha', None), ('Maxaafaa', None), ('tamaarissanau', 'PP'), ('giigissiyoobaappe', 'NC'), ('issuwaa', 'PP'), ('gidiShin', 'PP'), (',', None), ('he', 'PP'), ('oosoy', 'PP'), ('asay', 'PP'), ('ba', 'PP'), ('dosan', 'NC'), ('immiyo', 'PP'), ('miishshan', 'NC'), ('oosettees', 'VV'), ('.', 'AD'), ('geeshsha', 'PP'), ('maxaafaa', 'JJ'), ('taariketa', 'PP'), ('tamaariyo', 'PP'), ('maxafaa', 'JJ'), ('hagee', 'PP'), ('tumu', 'PP'),
```

('taariketa', 'PP'), ('oyqqida', 'PP'), ('maxaafa', 'JJ'), ('.', 'AD'), ('taariketi', 'PP'), ('ekettidoy', 'PP'), ('alamiyan', 'NC'), ('deâ\x80\x99iya', 'JJ'), ('ayba', 'PP'), ('maxaafatuppekka', 'JJ'), ('nashshettida', 'PP'), (',', None), ('geeshsha', 'PP'), ('maxaafaappe', 'NC'), ('.', 'AD'), ('ha', 'PP'), ('taariketi', 'PP'), ('xoossay', 'JC'), ('medhdhiyoogaa', 'JJ'), ('doommoosappe', 'NC'), ('hanno', 'PP'), ('gakkanaaShin', 'PP'), ('alamiyan', 'NC'), ('hanettidabata', 'PP'), ('yootoosona', 'VV'), ('.', 'AD'), ('eti', 'PP'), ('ubba', 'PP'), (',', None), ('xoossay', 'JC'), ('wodeppe', 'NC'), ('oottanaayyo', 'PP'), ('qaalaa', 'JJ'), ('gelidobatakka', 'PP'), ('yootoosona.', 'PP'), ('ha', 'PP'), ('maxaafay', 'JJ'), (',', None), ('geeshsha', 'PP'), ('maxaafan', 'NC'), ('deâ\x80\x99iya', 'JJ'), ('kumetta', 'NN'), ('qofay', 'JJ'), ('aybakko', 'NC'), ('qonccissees', 'VV'), ('.', 'AD'), ('etabaynne', 'NC'), ('eti', 'PP'), ('oottidobay', 'PP'), ('geeshsha', 'PP'), ('maxaafan', 'NC'), ('xaafettido', 'VV'), ('asatubaa', 'PP'), ('yootees.', 'PP'), ('i', 'PP'), (',', None), ('gannate', 'PP'), ('biittan', 'NC'), ('asay', 'PP'), ('merinau', 'JJ'), ('de', 'JJ'), ('anaadan', 'NC'), ('xoossay', 'JC'), ('giigissido', 'VV'), ('gita', 'PP'), ('hidootaabaakka', 'PP'), ('erissees', 'VV'), ('.', 'AD'), ('ha', 'PP'), ('maxaafan', 'NC'), ('116n', None), ('taariketi', 'PP'), ('deâ\x80\x99oosona', 'VV'), ('.', 'AD'), ('he', 'PP'), ('taariketi', 'PP'), ('hosppunan', 'NC'), ('shaahettidosona', 'VV'), ('.', 'AD'), ('haako', 'PP'), ('pala', 'PC'), ('poora', 'NC'), ('SHaahotu', 'PN'), ('ubbaayyo', 'PP'), ('doomettan', 'NC'), ('deâ\x80\x99iya', 'JJ'), ('issi', 'PP'), ('sinttay', 'PP'), (',', None), ('he', 'PP'), ('shaahuwaa', 'PP'), ('giddon', 'PP'), ('deâ\x80\x99iyabaa', 'JJ'), ('qanttan', 'NC'), ('yootees', 'VV'), ('.', 'AD'), ('taariketi', 'PP'), ('xaafettidoy', 'JC'), ('polettido', 'VV'), ('wodiyaa', 'PP'), ('maaraadaana', 'VV'), ('.', 'AD'), ('Hegee', None), (',', None), ('issi', 'PP'), ('hanettidabay', 'PP'), ('hara', 'NC'), ('taarikiyaara', 'NC'), ('gatti', 'PP'), ('xeelliyo', 'JC'), ('wode', 'PP'), (',', None), ('awude', 'PP'), ('polettidaakko', 'NC'), ('eranau', 'PP'), ('nena', 'VV'), ('maaddees', 'VV'), ('.', 'AD'), ('karetta', 'NN'), ('booray', 'PP'), ('cincatetta', 'NN'), ('gidenna', 'VV'), ('mala', 'JJ'), ('hara', 'NC'), ('haasayata', 'PP'), ('hanotetta', 'NN'), ('maatiyoosinne', 'NC'), ('saluwappe', 'NC'), ('gidiis', 'VV'), ('aayee', 'PP'), ('wano', 'PP'), ('guutta', 'PP'), ('qiiba', 'JJ'), ('maalado', 'JJ'), ('guuran', 'NC'), ('12', 'JN'), ('saaiyan', 'NC'), ('dendada', 'JJ'), ('baana', 'VV'), ('.', 'AD'), ('qiiba', 'JJ'), ('katta', 'JJ'), ('ekkada', 'PP'), ('baana', 'VV'), ('.', 'AD'), ('taaappe', 'NC'), ('simmin', 'PP'), ('baada', 'PP'), ('yaas.', 'PP'), ('kase', 'JJ'), ('nuussi', 'PP'), ('zino', 'JJ'), ('malatin', 'JJ'), (',', None), ('hayyana', 'VV'), ('aayyee', 'PP'), ('zo"o', 'JJ'), ("zo'o", 'JJ')……]

```
>>> random.seed(1234)

>>> train_set, test_set = train_test_split(sents_2, test_size=0.3)

>>> print(len(train_set))

54092

>>> print(len(test_set))

23183

>>> print(train_set[:2])

['doonaa', 'giyo']
```

```
>>> random.seed(1234)
>>> train_set, test_set = train_test_split(sents_2, test_size=0.1)
>>> print(len(train_set))
69547
>>> print(len(test_set))
7728
>>> print(train_set[:2])
['danddayssiyaagaa', 'bollaa']
>>> print(len(sents_2))
77275
```

Appendix II

Python codes for HMM approach Pos Tagging

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on
win32

Type "help", "copyright", "credits" or "license()" for more information.

import nltk, re, pprint

import numpy as np

import pandas as pd

import requests

import matplotlib.pyplot as plt

import seaborn as sns

import pprint, time

KeyboardInterrupt

import random

from sklearn.model_selection import train_test_split

from nltk.tokenize import word_tokenize

from nltk.corpus import hmmm

import gc

TRY_ZIPFILE_FIRXT = False

print(nltk.corpus.hmmm.tagged_words())

[('hachi', 'NN'), ('taani', 'PP'), ('uttada', 'VV'), ...]

nltk_data = list(nltk.corpus.nps_chat.tagged_posts())

print(nltk_data[:2])

[[('hachi', 'NN'), ('taani', 'PP'), ('uttada', 'VV'), ('pein', 'VV'), ('taassi', 'PC'), ('keehin', 'AD'), ('loisi',
'VV')], [('Geesha', 'JJ'), ('maxaafaa', 'NN'), ('taariketa', 'VV'), ('tamaariyo', 'VV'), ('maxaafaa', 'NN'),
('.', 'PN')]]
```

```
for sent in nltk_data[:2]:
  for tuple in sent:
    print(tuple)

('hachi', 'NN')
('taani', 'PP')
('uttada', 'VV')
('pein', 'VV')
('taassi', 'PC')
('keehin', 'AD')
('loisi', 'VV')
('Geesha', 'JJ')
('maxaafaa', 'NN')
('taariketa', 'VV')
('tamaariyo', 'VV')
('maxaafaa', 'NN')
('.', 'PN')
for sent in nltk_data[:101]:
  for tuple in sent:
    print(tuple)

('hachi', 'NN')
('taani', 'PP')
('uttada', 'VV')
('pein', 'VV')
('taassi', 'PC')
('keehin', 'AD')
('loisi', 'VV')
('Geesha', 'JJ')
('maxaafaa', 'NN')
('taariketa', 'VV')
('tamaariyo', 'VV')
('maxaafaa', 'NN')
('.', 'PN')
('Godattiya', 'NP')
('suufara', 'NN')
('gaccinna', 'AD')
('deddidaara', 'VV')
('soofiya', 'NN')
('gelawusu', 'VB')
('.', 'PN')
('O', 'PP')
('dosiya', 'JJ')
('laggeti', 'NC')
('ubbay', 'AX')
('Iira', 'PC')
('giya', 'NN')
('boosona', 'VV')
('.', 'PN')
('He', 'PP')
('sohuwan', 'PS')
('beettidaageeti', 'VC')
('uyoosonanne', 'VC')
("kaa'oosona", 'VV')
('.', 'PN')
('Dariyageeti', 'NN')
('qassi', 'CC')
('tanakka', 'PC')
('gattarkkii', 'II')
('goosona', 'VV')
('.', 'PN')
('Ta', 'PP')
('Sunttay', 'JJ')
('Anjja', 'NN')
('Anjjulla', 'NN')
('geetettees', 'VV')
('.', 'PN')
('Taani', 'PP')
('Sooddo', 'NN')
('Ambban', 'NN')
('oosanchchaanne', 'JC')
('issipetettaa', 'AD')
('poliya', 'NP')
('ooso', 'NN')
('keettaappe .', 'JC')
```

('yaas', 'VV')
('.', 'PN')
('Ha', 'PR')
('ooso', 'NN')
('keettan', 'JJ')
('taani', 'PP')
('poliyo', 'NP')
('biraa', 'NN')
('allaallee', 'JJ')
('alluwara', 'JN')
('gayttidaagaa', 'AD')
('gididi', 'JJ')
('erettees', 'VV')
('.', 'PN')
('Hagaara', 'PR')
('gayttidaagan', 'AD')
('alloy', 'JN')
('aybee', 'VV')
(',', 'PN')
('Ooni', 'PP')
('Oossi', 'PC')
('allii', 'VV')
('?', 'PN')
('aybissi', 'PC')
('allii', 'VV')
('?', 'PN')
('allin', 'VV')
('aybi', 'PP')
('merettii', 'AD')
('?', 'PN')
('allana', 'VV')
('xayin', 'AD')
('gakkiya', 'JJ')
('metoy', 'NN')
('aybee', 'PP')
('?', 'PN')
('giya', 'PP')
('qopatun', 'NC')
('taani', 'PP')
('intessi', 'PC')
('haasayiyoobaa', 'NC')
('loyttidi', 'AD')
('siyite', 'VV')
('.', 'PN')
('Alluwaa', 'NN')
('giyoogee', 'JJ')
('asay', 'NN')
('asaara', 'NC')
('issippe', 'JC')
("de'iyo", 'JJ')
('duussaa', 'NN')
('giddon', 'AD')
('oonne', 'PP')
('qoppennan', 'AD')
('metoy', 'NN')
('kiyidosaa', 'AD')
('gakkiyoo', 'AD')
('hanotaa', 'VV')
('.', 'PN')
('Hegaassi', 'PC')
('gattiyabay', 'AD')
('qassi', 'CC')
('akeekennan', 'AD')
('siyettiya', 'JJ')
('waasuwanne', 'NC')
('xurumbettiya', 'NP')
('xurumbaa', 'NN')
(',', 'PN')
('qoppennan', 'AD')
('siyettiya', 'NP')
('yeehuwa', 'NN')
(',', 'TO')
('Hegaappe', 'PC')
('dendidaagan', 'AD')
('naatoy', 'NN')

('cimaassi', 'NC')
(',', 'PN')
('shooroy', 'NN')
('shooruwassi', 'NP')
(',', 'PN')
('lagee', 'NN')
('laggiyassi', 'NC')
(',', 'PN')
('zaree', 'NN')
('zariyassi', 'PC')
('allees', 'VV')
('.', 'PN')
('Allana', 'NN')
('yiya', 'JJ')
('asi', 'NN')
('wocamaanne', 'NC')
('waasuwappe', 'NC')
('attin', 'PR')
('hanidabaa', 'JJ')
('erenna', 'JJ')
('gishshawu', 'CC')
('ba', 'PP')
('kushiyan', 'NC')
('gayttidabaa', 'AD')
('ekkinne', 'AD')
('woxxi', 'AD')
('agees', 'VV')
('.', 'PN')
('woxxiiddikka', 'AD')
('taykki', 'II')
(';', 'PN')
('laa', 'PP')
('aybee', 'PP')
(',', 'PN')
('saro', 'NN')
('gidennee', 'AD')
('?', 'PN')
('yaagiiddi', 'AD')
('issoy', 'JN')
('issuwa', 'JN')
('dentettiyo', 'AD')
('oosuwa', 'AD')
('oottees', 'VV')
('.', 'PN')
('Hagaa', 'PP')
('maaran', 'AD')
('waasoy', 'NN')
('merettidosaa', 'JJ')
('gakkiya', 'AD')
('alloy', 'NN')
('gakkiyo', 'AD')
('wode', 'PR')
('waretay', 'NN')
('merettidosaa', 'JJ')
('gidikko', 'AD')
('warettiyageetu', 'NC')
('giddo', 'PR')
('gelidi', 'VV')
('ashshiyo', 'AD')
('oosuwa', 'NN')
('oottees', 'VV')
(',', 'PN')
('a', 'PP')
('shociyaagaa', 'JJ')
('kushiyappe', 'NC')
('dullaa', 'NN')
(',', 'PN')
('qanxxiyagaappenne', 'JJ')
('caddiyagaa', 'JJ')
('kushiyappe', 'NP')
('mashshaanne', 'NC')
('tooraa', 'NN')
(',', 'PN')
('oliyaagaappe', 'JJ')
('qassi', 'CC')

('shuchaa', 'NN')
('wotti', 'AD')
('ekees', 'VV')
('.', 'PN')
('Hegee', 'PP')
('haniyo', 'AD')
('wode', 'PR')
('shemppoy', 'NN')
('hayqennan', 'AD')
('attees', 'VV')
(';', 'PN')
('aypee', 'NN')
('achchay', 'NN')
('qohettiyosaappe', 'AD')
('palahees', 'VV')
(';', 'PN')
('shooroy', 'NN')
(',', 'PN')
('heeray', 'NN')
('mettuwappenne', 'NC')
('palaappe', 'NC')
('shempees', 'VV')
('.', 'PN')
('Balggoy', 'NN')
('wolqaamiyo', 'JJ')
('layttan', 'NN')
('iray', 'NN')
('dariyoogaappe', 'JJ')
('denddidaagan', 'AD')
('heeran', 'AD')
("de'iya", 'JJ')
('haattay', 'NN')
('kixxees', 'VV')
('.', 'PN')
('Ha', 'VB')
('kixxida', 'PC')
('haattay', 'AD')
('giyaappe', 'CC')
('gidin', 'AD')
('mandaraappe', 'JJ')
('yiya', 'NC')
('asaa', 'NP')
('miyo', 'NN')
('wode', 'PN')
("de'ees", 'NN')
('.', 'PN')
('ziizziyaara', 'NC')
('walaqasiyo', 'AD')
(';', 'PN')
('xaaxi', 'PR')
('waaxiyo', 'PR')
('wode', 'PR')
('aybatettanne', 'PC')
('alluwan', 'NN')
('polettiya', 'AD')
('allaalleta', 'NC')
('loyttidi', 'AD')
('siyideta', 'VV')
('.', 'PN')
('Oonne', 'PP')
('qoppenna', 'JJ')
('metoy', 'NN')
('merettosaa', 'AD')
('woxxidi', 'VV')
('gakkiyo', 'AD')
('hanotay', 'VV')
('ayba', 'PP')
('geetettii', 'AD')
('?', 'PN')
('Metootaa', 'NC')
('Maadiyoogaa', 'VV')
('Meto', 'NN')
('giyogee', 'AD')
('issi', 'JN')
('asa', 'NN')

('abbiyan', 'NC')
('poletennan', 'AD')
('ixxidabaanne', 'JJ')
('he', 'PP')
('asi', 'NN')
('asappe', 'NC')
('garssaara', 'AD')
('aadhanaadan', 'AD')
('oottidabaa', 'VV')
('.', 'PN')
('Hegaa', 'PP')
('maaran', 'PR')
('gakkida', 'AD')
('un’uwappe', 'NC')
('kiyana', 'VV')
('giin', 'AD')
('metoy', 'VV')
('xibbuwan-tuccido', 'JJ')
('asa', 'NN')
('metootanchchaa', 'AD')
('geetettees', 'VV')
('.', 'JJ')
('Asay', 'NN')
('hiyyeesatettan', 'NC')
(',', 'PN')
('sahuwan', 'NC')
(',', 'PN')
('namisan', 'NC')
(',', 'PN')
('olan', 'NC')
(',', 'PN')
('zariya', 'NN')
('dabbuwa', 'NN')
('xayuwan', 'AD')
('metootana', 'AD')
('dandayees', 'VV')
('.', 'PN')
('Oosoy', 'NN')
('xayin', 'AD')
('metootiyagaassi', 'JJ')
('nu', 'PP')
('sooni', 'NN')
("de'iya", 'JJ')
('gidaappe', 'NC')
('shaakkidi', 'AD')
('immidi', 'AD')
('oosuwa', 'NN')
('merissikko', 'JC')
('a', 'PP')
('metoy', 'NN')
('hayse', 'NN')
('kiyees', 'VV')
('.', 'PN')
('Issitoo', 'JN')
('issitoo', 'JN')
('laggetiya', 'JJ')
(',', 'PN')
('dabbotiyanne', 'JJ')
('wozanabaa', 'NN')
('zoretissiyo', 'JJ')
('asi', 'NN')
('xayin', 'AD')
('metootiyanne', 'NC')
('iitaban', 'NC')
('geliya', 'AD')
('asaykka', 'NC')
("de'ees", 'VV')
('.', 'PN')
('dabboy', 'NN')
('dabbuwa', 'NN')
(',', 'PN')
('laggee', 'NN')
('bari', 'PP')
('laggiya', 'NN')
('kanchiyawu', 'PR')

('gakkidi', 'AD')
('laggeenne', 'NC')
('dabboy', 'NN')
('baynnaagaa', 'AD')
('pali-poorawu', 'II')
('gattiyo', 'AD')
('metuwa', 'NN')
('pagalana', 'AD')
('bessees', 'VV')
('.', 'PN')
('Bonchidi', 'AD')
('Bonchetteettees', 'VV')
('Daani', 'NN')
('bari', 'PP')
('aawaassinne', 'NC')
('bari', 'PP')
('aayeessi', 'NC')
('kushe', 'NN')
('toho', 'NN')
('phalqqida', 'VV')
('bayra', 'NN')
('.', 'PN')
('Hegaappe', 'PC')
('dendidaagan', 'PR')
('eta', 'PP')
('soon', 'NN')
(';', 'PN')
('woxxi-qaaxxidabawu', 'AD')
('a', 'PP')
('xeesoosona', 'VV')
('.', 'PN')
('Bantta', 'PC')
('soo', 'NN')
('asaappenne', 'NC')
('luxxissiyageetuppe', 'NC')
('marccuwa', 'NN')
('shiishidi', 'AD')
('biiriya', 'NN')
('dawutaraa', 'NN')
('hara', 'JJ')
('hiyeesa', 'JJ')
('asata', 'NC')
('i', 'PP')
('maaddees', 'VV')
('.', 'PN')
('ne-hayllawu', 'II')
('hagaa', 'PR')
('yelida', 'JJ')
('aayyiya', 'NN')
('yaagosona', 'VV')
('.', 'PN')
('Ashkanne', 'NC')
('kuntti', 'NN')
('sooddo', 'NN')
('ambba', 'NN')
('Ganddabba', 'NN')
('shuchan', 'NN')
('machchonne', 'NC')
('azina', 'NN')
('gididi', 'AD')
("de'oosona", 'VV')
('.', 'PN')
('So', 'NN')
('bagga', 'PR')
('toohoy', 'NN')
('daro', 'AD')
('baggi', 'PR')
('i', 'PP')
('hashiyan', 'NC')
('kundido', 'AD')
('ashkka', 'NN')
('kaseegaappe', 'PR')
('woxxi-qaaxxaban', 'AD')
('puuriyogaanne', 'JC')
('gancariya', 'AD')

('dariyaro', 'AD')
('gidaasu', 'VV')
('.', 'PN')
('kase', 'PR')
('soo', 'NN')
('geliiddi', 'AD')
('Ashkee', 'NN')
('giyo', 'AD')
('wode', 'PR')
('dooni-gaanawu', 'II')
('yee', 'VV')
(',', 'PN')
('Hayqqida-abbiyan', 'II')
('yee', 'VV')
(',', 'PN')
('waanadii', 'AD')
(',', 'PN')
('ayssi', 'PP')
('xeesay', 'VV')
('?', 'PN')
('ne', 'PP')
('ay', 'PP')
('maaddiyoosaara', 'AD')
('daaray', 'AD')
('yaagaasu', 'VV')
('.', 'PN')
('Hallittay', 'NN')
('ooggiichchin', 'AD')
('hanniira', 'PC')
('aadhdharkina', 'AD')
('geettees', 'VV')
('.', '.')
('Daamoota', 'NN')
('pullaasa', 'NN')
('allaanan', 'NN')
('malddaaye', 'NN')
('Birillo', 'NN')
('geetettiya', 'AD')
('luxanchchiya', 'NN')
("de'awusu", 'VV')
('.', 'PN')
('Birillissi', 'NC')
('meraa', 'NN')
("lo'otettikka", 'JJ')
('keehin', 'AD')
('meto', 'NN')
('gididabaappe', 'JC')
('issuwa', 'JN')
('.', 'PN')
('kaalliya', 'PR')
('Nabbabuwa', 'NN')
('Tololayada', 'AD')
('Nabbaba', 'VV')
('Humbbo', 'NN')
('Allaanaa', 'NN')
('giddon', 'PR')
('kollojjiya', 'NN')
('gelada', 'AD')
('luxa', 'AD')
('anjjettida', 'VV')
('anjja', 'NN')
('ishalo', 'NN')
('geetettiya', 'AD')
('yelagiya', 'NC')
("de'awusu", 'VV')
('.', 'PN')
('kaalliya', 'PR')
('Nabbabuwa', 'NN')
('Tololayada', 'AD')
('Nabbaba', 'VV')
('Humbbo', 'NN')
('Allaanaa', 'NN')
('giddon', 'PR')
('kollojjiya', 'NN')
('gelada', 'AD')

('luxa', 'AD')
('anjjettida', 'VV')
('anjja', 'NN')
('ishalo', 'NN')
('geetettiya', 'AD')
('yelagiya', 'NC')
("de'awusu", 'VV')
('.', 'PP')
('Ishala', 'NN')
('issi', 'JN')
('ooso', 'JJ')
('keettay', 'NN')
('ooratta', 'JJ')
('oosancha', 'NN')
('ekkanaasi', 'AD')
('erissuwa', 'NN')
('kessidoogaa', 'AD')
("be'aasu", 'VV')
('.', 'PN')
('Ha', 'PP')
('erissuwa', 'AD')
("be'ida", 'VV')
('Ishala', 'NN')
('waaciwaacaydda', 'AD')
('soo', 'NN')
('puttu', 'AD')
('gosaarakka', 'VV')
(',', 'PN')
('Ta', 'PP')
('aayilee', 'NN')
('taassi', 'PC')
('kaamiya', 'NN')
('toggada', 'AD')
('kolloojiya', 'NN')
('biyo', 'AD')
('marccuwa', 'NN')
('imma', 'VV')
('yagaasu', 'AD')
('.', 'PN')
('Ishali', 'NN')
('aayyiya', 'NN')
('ta', 'PP')
("na'ee", 'NN')
("lo'o", 'VV')
('gidenneeye', 'AD')
('?', 'PN')
('Nena', 'PP')
('luxada', 'AD')
('kiyido', 'JJ')
('kollojiya', 'NN')
('guyye', 'PR')
('zaariyabi', 'NN')
('aybee', 'PP')
('?', 'PN')
('yaagaasu', 'VV')
('.', 'PN')
('Ishala', 'NN')
('Zaarada', 'PR')
('ta', 'PP')
('aayiloo', 'NN')
('guyye', 'PR')
('simmanaassa', 'AD')
('gidenna', 'VV')
(';', 'PN')
('oosuwa', 'NN')
('gelanaassi', 'AD')
('erissoy', 'NN')
('kiyidaagaa', 'JJ')
("be'ido", 'AD')
('gishawu', 'CC')
('taani', 'PP')
('luxettaa', 'NN')
('wursidoogaa', 'AD')
('boxooxissiya', 'AD')
('markka', 'JJ')

('woraqataa', 'NN')
('ekkanawu', 'AD')
('baanaassa', 'VV')
('yaagaasu', 'VV')
('.', 'PN')
('Ishali', 'NN')
('aayyiyakka', 'NC')
("sa'an", 'NN')
('gulbataa', 'AD')
('lichi', 'JJ')
('gosaara', 'AD')
('xoossoo', 'NN')
('hagee', 'PP')
('ne', 'PP')
('wolqqa', 'NN')
('!', 'NN')
('hay', 'II')
('tanakka', 'NC')
('hagawu', 'PR')
('gattadii', 'AD')
('!', 'PN')
('giidaara', 'AD')
('dancuwa', 'NN')
('birshosaara', 'AD')
('iyyo', 'PP')
('marccuwa', 'NN')
('imma', 'VV')
('aggaasu', 'VV')
('.', 'PN')
('Ishalakka', 'NN')
('bari', 'PP')
('aayeeppe', 'NC')
('marccuwa', 'NN')
('ekkosaara', 'AD')
('kaamiya', 'NN')
('toggada', 'AD')
('kollojiya', 'NN')
('cuchchi', 'NN')
('melennan', 'AD')
('gakka', 'VV')
('aagaasu', 'VV')
('.', 'PN')
('A', 'PP')
('kolloojjiya', 'NN')
('gakkosaara', 'AD')
('biiruwa', 'NN')
('geliyo', 'AD')
('wode', 'PR')
('biiruwa', 'NN')
('penggee', 'NN')
('dooyaShin', 'AD')
('giddon', 'PR')
('kana', 'NN')
('shocciyabi', 'AD')
('baawa', 'VV')
('.', 'PN')
('Ishalissi', 'NC')
('haniyobay', 'AD')
('kirqqi', 'II')
('gin', 'CC')
('pude', 'PR')
('xeelliyo', 'AD')
('wode', 'AD')
('silkke', 'NN')
('payduwa', 'NN')
('xaafidi', 'VV')
('kaqqi', 'AD')
('wottidoogaa', 'AD')
('demmidaara', 'AD')
('shoca', 'VV')
('agaasu', 'VV')
('.', 'PN')
('Silkkee', 'NN')
('xeesees', 'VV')
('Shin', 'CC')

('denttennan', 'AD')
('ixxiis', 'VV')
('.', 'PN')
('Ishalakka', 'NC')
('usa', 'II')
('hachchi', 'PR')
('ta', 'PP')
('allaalliyo', 'NN')
('polissennan', 'AD')
('hasooppe', 'NC')
('kiyikke', 'VV')
('giidaara', 'AD')
('penggen', 'NC')
('uttaaggada', 'AD')
('naaguwa', 'VV')
('sheekkaasu', 'VV')
('.', 'PN')
('Utettay', 'JJ')
('palin', 'II')
('zaarettada', 'PR')
('silkkiya', 'NN')
('shocin', 'AD')
('denttidi', 'VV')
('shiiquwan', 'NN')
("de'ays", 'VV')
(';', 'PN')
('laaxaappe', 'NC')
('guyyiyan', 'PR')
('ya', 'AD')
('yaagiis', 'VV')
('.', 'PN')
('Ishalakka', 'NC')
('hayya', 'II')
('!', 'PN')
('Hagee', 'PP')
('laaxappe', 'NC')
('guyyiyawu', 'PR')
('keeridaagee', 'AD')
('hodde', 'JJ')
('bari', 'PP')
('keerido', 'AD')
('wodiyan', 'AD')
('biiruwa', 'NN')
('gelees', 'VV')
('gaada ', 'AD')
('laaxaa', 'NN')
('qaphali', 'JJ')
('oottosaara', 'AD')
('aggada', 'AD')
('yiidaara', 'VV')
('biiruwa', 'NN')
('penggen ', 'NN')
('utta', 'VV')
('aggaasu', 'VV')
('.', 'PN')
('Biyagee', 'AD')
('yiyagee', 'AD')
('A', 'PP')
('malatin', 'JJ')
('phuluqqattiShin', 'NN')
('away', 'NN')
('lekkeme’i', 'AD')
('aggiis', 'VV')
('.', 'PN')
('Biyagee', 'AD')
('yiyagee', 'AD')
('A', 'PP')
('malatin', 'JJ')
('phuluqqattiShin', 'NN')
('away', 'NN')
('lekkeme’i', 'AD')
('aggiis', 'VV')
('.', 'PN')
('Issi', 'JN')
('haniyobay', 'AD')

('leelin', 'AD')

('zaarettadakka ', 'AD')

('silkkiya', 'NN')

('shocin', 'VV')

('beeruwa', 'NN')

('payduwa ', 'NN')

('hosppunan', 'JN')

("de'iyageeta", 'PP')

('baada', 'VV')

('haasayissa', 'VV')

('taani', 'PP')

('hachchi', 'AD')

('gelikke', 'AD')

('yaagiis', 'VV')

('.', 'PN')

('Ha'ikka', 'PR')

('sheniya', 'NN')

('hayqqada', 'NC')

('he', 'PP')

('beeruwa', 'NN')

('biyo', 'AD')

('wode', 'AD')

('biiroy', 'NN')

('dooya', 'AD')

('Shin', 'CC')

('issi', 'JN')

('asikka', 'NC')

('baawa', 'VV')

('.', 'PN')

('Ishala', 'NN')

('silkkiya', 'NN')

('shocciyo', 'AD')

('wode', 'AD')

('allaallee', 'NN')

('xeelliyo', 'AD')

('izaaya', 'PP')

('taani', 'PP')

('yeeho', 'NN')

('baas', 'VV')

(';', 'PN')

('biiruwa', 'NN')

('payduwa', 'NN')

('heezzaa', 'JN')

('baada', 'AD')

('ebela', 'PP')

('oycha', 'VV')

(';', 'PN')

('naqaashay', 'NN')

('a', 'PP')

('matan', 'PR')

("de'ees", 'AD')

('yagaasu', 'VV')

('.', 'PN')

('Ishalakka', 'NC')

('yaanne', 'PR')

('haanne', 'PR')

('kiirimmishin', 'AD')

('daafuray', 'NN')

('alana', 'AD')

('hanishin', 'AD')

('baada', 'AD')

('ta', 'PP')

('ishawu', 'NN')

('ne', 'PP')

('ooso', 'NN')

('miiriya', 'NN')

('ta', 'PP')

('naqaashay', 'AD')

('ne', 'PP')

('biiruwan', 'NN')

("de'ees", 'VV')

(';', 'PN')

('gaada', 'AD')

('liiqo', 'JJ')

('qaalan', 'NN')

('oychaasu', 'VV')
('.', 'PN')
('o', 'PP')
('xoqqu', 'PR')
('gi', 'AD')
('xeelli', 'NN')
('simmidi', 'PR')
('komppiiteriya', 'NN')
('miqiqqilliiddi', 'AD')
('xomppee', 'NN')
('baawa', 'VV')
(';', 'PN')
('wontto', 'AD')
('ya', 'AD')
('yaagiis', 'VV')
('.', 'PN')
('yaatin', 'JJ')
('ishalissi', 'NC')
('ayfiyappe', 'NC')
('afuttay', 'NN')
('guppi', 'AD')
('guppi', 'AD')
('kiyishin', 'AD')
('namisan', 'NC')
(',', 'PN')
('saamuwan', 'NC')
("kaa'ettada", 'AD')
('yiido', 'VV')
('allaalliyanne', 'NC')
('polennan', 'AD')
('bantasoo', 'PC')
('guyye', 'PR')
('simmaasu', 'VV')
('.', 'PN')
('Ooso', 'NN')
('wodiyan', 'JJ')
('ooso', 'NN')
('sohuwan', 'PR')
('woykko', 'CC')
('biiruwan', 'NN')
('uttenaagee', 'AD')
('yeehuwa', 'NN')
('cayetaa', 'NN')
('qoriyooge', 'AD')
(';', 'PN')
('silkkiya', 'NN')
('shocin', 'AD')
('wora', 'PR')
('malaatiyoogee', 'AD')
('issi', 'NN')
('allaalliya', 'AD')
('polissanawu', 'CC')
('haahuwappe', 'JN')
('gidin', 'NC')
('mataappe', 'VV')
('yiida', 'AD')
('izaawu', 'AD')
('bollaa', 'NN')
(',', 'AD')
('wolqqaa', 'CC')
(',', 'JN')
('wodiyanne', 'NC')
('marccuwa', 'VV')
('h.h.m', 'PN')
('qohuwaa', 'PN')
('gattiyo', 'AD')
('gishshawu', 'AD')
('ooso', 'NN')
('wodiya', 'AD')
('suure', 'CC')
("go'ettana", 'JN')
('bessees', 'NC')
('.', 'PN')
('Azallay', 'PP')
('asabaa', 'NC')

('amottiddi', 'AD')
('atti', 'AD')
('aggees', 'VV')
('Wolaytta', 'NN')
('Moottan', 'PR')
('Bolooso', 'NN')
('Bombbe', 'NN')
('allaanan', 'NC')
('issi', 'JN')
('keehi', 'JJ')
('mino', 'JJ')
('asappe', 'NC')
('yelettida', 'AD')
('shaaga', 'JJ')
('shola', 'NN')
('geettettiya', 'AD')
('asi', 'NN')
("de'ees", 'VV')
('.', 'PN')
('Ha', 'PP')
('asassi', 'NC')
('bari', 'PP')
('aawaappe', 'NC')
('laattido', 'AD')
('aaho', 'JJ')
('gadee', 'NN')
(',', 'PN')
('ala', 'JJ')
('uuttay', 'NN')
(',', 'PN')
('gaasho', 'JJ')
('keettay', 'NN')
(',', 'PN')
('dirssa', 'JJ')
('miizzay', 'NN')
(',', 'PN')
('konssa', 'JJ')
('mittay', 'NN')
(',', 'PN')
('xalbuqa', 'JJ')
('dooreenne', 'NC')
('gootaray', 'NN')
("de'ees", 'VV')
('.', 'PN')
('Hegaa', 'PP')
('xalla', 'CC')
('gidennan', 'AD')
('aaho', 'JJ')
('karee', 'NN')
(',', 'PN')
('narkkakka', 'JJ')
('kuway', 'NN')
(',', 'PN')
('woxxi', 'CC')
('qaaxxidaban', 'AD')
('ekkiyoba', 'AD')
('ubbay', 'PP')
('kumi', 'AD')
('palahido', 'AD')
('asa', 'NN')
('.', 'PN')
('Ha', 'PP')
('ubba', 'PP')
('gidaa', 'NN')
('bari', 'PP')
('aawaappe', 'NC')
('laattida', 'JJ')
('shaaga', 'NN')
('sholi', 'NN')
('laatti', 'AD')
('demmido', 'AD')
('aqota', 'NN')
('minni', 'JJ')
('oottidi', 'VV')
('yaa', 'AD')

('yaa', 'AD')
('aassanawunne', 'AD')
('dichanawu', 'AD')
('dandaybeynna', 'VV')
('.', 'PN')
('Beettidabaa', 'JJ')
('ubbaa', 'PP')
('bayzzi', 'VV')
('bayzzi', 'VV')
('miyoogaa', 'VV')
(',', 'PN')
('uyiyoogaa', 'VV')
(',', 'PN')
('coo', 'AD')
('alleeqiyoogaanne', 'VV')
('wogettiyoogaa', 'VV')
('doommiis', 'VV')
('.', 'PN')
("De'iyaban", 'AD')
('otoretteesippe', 'II')
('attin', 'CC')
('kunddidikka', 'AD')
('biittaa', 'NN')
('oyqqanawu', 'AD')
('koyenna', 'VV')
('azalla', 'JJ')
('asa', 'NN')
('.', 'PN')
('Hegaappe', 'PC')
('denddidaagan', 'AD')
('oyqqobay', 'VV')
('yedhdhobay', 'VV')
('qammaappe', 'NC')
('qamman', 'NC')
('herddi', 'NN')
('herddi', 'NN')
('biyo', 'AD')
('hanotay', 'NN')
('merettiis', 'VV')
('.', 'PN')
('Kattay', 'NN')
('dooriyappenne', 'NC')
('gootaraappe', 'NC')
('wurin', 'AD')
(',', 'PN')
('alay', 'NN')
('miishininne', 'VV')
('bayzzishin', 'VV')
('tooshettin', 'VV')
('.', 'PN')
('miizzay', 'NN')
('zadaluwappenne', 'NC')
('gaxaataappe', 'NC')
('paccin', 'AD')
('namisay', 'NN')
(',', 'PN')
('kalloy', 'NN')
('kabbashay', 'NN')
(',', 'PN')
('acoy', 'NN')
('galee', 'NN')
('a', 'PP')
('soo', 'NN')
('geliyoogee', 'VV')
('akeekettiis', 'VV')
('.', 'PN')
('Hagaadan', 'PC')
("de'oy", 'NN')
('moorettido', 'AD')
('sholi', 'NN')
("ha'ikka", 'PR')
('eqqi', 'VV')
('ekkidi', 'AD')
('bana', 'PP')
('giigissiyo', 'AD')

('kahaa', 'NN') ('qanxxi', 'VV') ('kushee', 'NN')

('demmennan', 'AD') ('qanxxi', 'VV') ("lee'iichchidoogaa", 'AD')

('asa', 'NN') ('bayzziyogaa', 'VV') ("be'ida", 'AD')

('soo', 'NN') ('giddo', 'PR') ('kase', 'PR')

('hemettiyoge', 'AD') ('geliis', 'VV') ('A', 'PP')

(',', 'PN') ('.', 'PN') ('bonchchiya', 'JJ')

('asabaa', 'PP') ('A', 'PP') ('laggeenne', 'NC')

('qaaqqatiyoogee', 'AD') ('soo', 'NN') ('dabboy', 'NN')

('erettiis', 'VV') ('wogibaynna', 'JJ') ("ha'i", 'PR')

('.', 'PN') ('pacaynne', 'NC') ('appe', 'PC')

('Ubba', 'PP') ('namisay', 'NN') ('haahuwara', 'PR')

('ixxosan', 'JJ') ('lawuhu', 'AD') ('baqatiddi', 'AD')

('hani', 'AD') ('giis', 'VV') ('beettiis', 'VV')

('salidi', 'AD') ('.', 'PN') ('.', 'PN')

('biittaa', 'NN') ('Shola', 'NN')

```
train_set,test_set =train_test_split(nltk_data,train_size=0.90,test_size=0.10,random_state = 101)

train_tagged_words = [ tup for sent in train_set for tup in sent ]

test_tagged_words = [ tup for sent in test_set for tup in sent ]

print(len(train_tagged_words))
```

2367

```
print(len(test_tagged_words))
```

283

```
print(len(nltk_data))
```

228

```
tags = {tag for word,tag in train_tagged_words}

print(len(tags))
```

22

```
from nltk.corpus import hmmm
import gc
TRY_ZIPFILE_FIRXT = False
print(nltk.corpus.hmmm.tagged_words())
[('hachi', 'NN'), ('taani', 'PP'), ('uttada', 'VV'), ...]
nltk_data = list(nltk.corpus.hmmm.tagged_posts())
print(nltk_data[:2])
[[('hachi', 'NN'), ('taani', 'PP'), ('uttada', 'VV'), ('pein', 'VV'), ('taassi', 'PC'), ('keehin', 'AD'), ('loisi', 'VV')], [('Geesha', 'JJ'), ('maxaafaa', 'NN'), ('taariketa', 'VV'), ('tamaariyo', 'VV'), ('maxaafaa', 'NN'), ('.', 'PN')]]
train_set,test_set =train_test_split(nltk_data,train_size=0.90,test_size=0.10,random_state = 101)
train_tagged_words = [ tup for sent in train_set for tup in sent ]
test_tagged_words = [ tup for sent in test_set for tup in sent ]
print(len(train_tagged_words))
2367
print(len(test_tagged_words))
283
print(len(nltk_data))
228
KeyboardInterrupt
tags = {tag for word,tag in train_tagged_words}
print(len(tags))
18
from nltk.corpus import hmmm
import gc
```

```
TRY_ZIPFILE_FIRXT = False
```

```
print(nltk.corpus.nps_chat.tagged_words())
```

```
[('hachi', 'NN'), ('taani', 'PP'), ('uttada', 'VV'), ...]
```

```
[('hachi', 'NN'), ('taani', 'PP'), ('uttada', 'VV'), ...]
```

```
[('hachi', 'NN'), ('taani', 'PP'), ('uttada', 'VV'), Ellipsis]
```

```
nltk_data = list(nltk.corpus.hmmm.tagged_posts())
```

```
print(nltk_data[:2])
```

```
[[('hachi', 'NN'), ('taani', 'PP'), ('uttada', 'VV'), ('pein', 'VV'), ('taassi', 'PC'), ('keehin', 'AD'), ('loisi', 'VV')], [('Geesha', 'JJ'), ('maxaafaa', 'NN'), ('taariketa', 'VV'), ('tamaariyo', 'VV'), ('maxaafaa', 'NN'), ('.', 'PN')]]
```

```
train_set,test_set =train_test_split(nltk_data,train_size=0.90,test_size=0.10,random_state = 101)
```

```
train_tagged_words = [ tup for sent in train_set for tup in sent ]
```

```
test_tagged_words = [ tup for sent in test_set for tup in sent ]
```

```
print(len(train_tagged_words))
```

```
2367
```

```
print(len(test_tagged_words))
```

```
283
```

```
print(len(test_tagged_words+train_tagged_words))
```

```
2650
```

```
tags = {tag for word,tag in train_tagged_words}
```

```
print(len(tags))
```

```
17
```

```
print(tags)
```

```
{'JC', 'II', 'PN', 'VB', 'AD', 'NC', 'JJ', 'PR', 'CC', 'PC', 'CNC', 'PRC', 'VV', 'ONC', 'PP', 'NN', 'ON'}
```

```
vocab = {word for word,tag in train_tagged_words}
```

```
def word_given_tag(word, tag, train_bag = train_tagged_words):
```

```
    tag_list = [pair for pair in train_bag if pair[1]==tag]
    count_tag = len(tag_list)#total number of times the passed tag occurred in train_bag
    w_given_tag_list = [pair[0] for pair in tag_list if pair[0]==word]
    count_w_given_tag = len(w_given_tag_list)
    return (count_w_given_tag, count_tag)


def t2_given_t1(t2, t1, train_bag = train_tagged_words):
     tags = [pair[1] for pair in train_bag]
     count_t1 = len([t for t in tags if t==t1])
     count_t2_t1 = 0
     for index in range(len(tags)-1):
         if tags[index]==t1 and tags[index+1] == t2:
             count_t2_t1 += 1
     return (count_t2_t1, count_t1)
   tags_matrix = np.zeros((len(tags), len(tags)), dtype='float32')
for i, t1 in enumerate(list(tags)):
     for j, t2 in enumerate(list(tags)):
         tags_matrix[i, j] = t2_given_t1(t2, t1)[0]/t2_given_t1(t2, t1)[1]
        print(tags_matrix)
[[0.         0.         0.         0.         0.         0.
  0.         0.33333334 0.         0.         0.         0.
  0.6666667  0.         0.         0.         0.         ]
 [0.         0.06896552 0.10344828 0.         0.20689656 0.10344828
  0.06896552 0.10344828 0.10344828 0.06896552 0.         0.
  0.10344828 0.         0.         0.06896552 0.         ]
```

[0. 0.0141844 0.0035461 0.0035461 0.07801419 0.09574468
 0.05673759 0.05319149 0.0106383 0.04255319 0. 0.0070922
 0.0248227 0. 0.23049645 0.36524823 0.0106383]
[0. 0. 0. 0. 0. 0.
 0. 0. 0. 1. 0. 0.
 0. 0. 0. 0. 0.]
[0.00431965 0.01727862 0.05399568 0. 0.2138229 0.03455723
 0.06047516 0.04103672 0.03671706 0.00647948 0. 0.00431965
 0.23974082 0. 0.06263499 0.21814255 0.00647948]
[0. 0.02013423 0.08724833 0. 0.2080537 0.12080537
 0.06711409 0.05369128 0.02684564 0. 0. 0.00671141
 0.12080537 0.00671141 0.04697987 0.22147651 0.01342282]
[0. 0.00680272 0.02721088 0. 0.10884354 0.10204082
 0.06122449 0. 0.06122449 0. 0. 0.01360544
 0.10884354 0. 0.02721088 0.4829932 0.]
[0. 0.01020408 0.06122449 0. 0.23469388 0.
 0.03061225 0.09183674 0.01020408 0.03061225 0. 0.
 0.20408164 0. 0.05102041 0.2755102 0.]
[0. 0. 0.01785714 0. 0.19642857 0.05357143
 0.01785714 0.05357143 0.03571429 0.03571429 0. 0.01785714
 0.10714286 0. 0.25 0.19642857 0.01785714]
[0. 0.08571429 0. 0. 0.37142858 0.02857143
 0.02857143 0.11428571 0.02857143 0. 0. 0.
 0.08571429 0. 0.08571429 0.17142858 0.]
[0. 0. 0. 0. 0. 0.

```
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         1.         0.        ]
[0.         0.         0.         0.         0.         0.125
 0.25       0.         0.         0.         0.         0.
 0.         0.         0.         0.375      0.25      ]
[0.         0.         0.5062893  0.         0.11949685 0.03773585
 0.0408805  0.00943396 0.00628931 0.01886792 0.00314465 0.
 0.1163522  0.         0.03773585 0.10377359 0.        ]
[0.         0.         0.         0.         0.         0.
 0.25       0.         0.         0.         0.         0.
 0.25       0.25       0.25       0.         0.        ]
[0.         0.00529101 0.01587302 0.         0.20634921 0.11111111
 0.08465608 0.02645503 0.02116402 0.01587302 0.         0.
 0.06878307 0.         0.08994709 0.34391534 0.01058201]
[0.00176678 0.01060071 0.11307421 0.         0.2862191  0.05477032
 0.07420494 0.04946997 0.01766784 0.00530035 0.         0.
 0.14134276 0.00353357 0.05123675 0.18551236 0.00530035]
[0.         0.         0.05555556 0.         0.16666667 0.05555556
 0.16666667 0.         0.         0.         0.         0.
 0.05555556 0.         0.11111111 0.2777778  0.11111111]]
>>> tags_df = pd.DataFrame(tags_matrix, columns = list(tags), index=list(tags))
>>> print(tags_df)
          JC        II        PN  ...        PP        NN        ON
JC  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.000000
II  0.000000  0.068966  0.103448  ...  0.000000  0.068966  0.000000
```

PN 0.000000 0.014184 0.003546 ... 0.230496 0.365248 0.010638

VB 0.000000 0.000000 0.000000 ... 0.000000 0.000000 0.000000

AD 0.004320 0.017279 0.053996 ... 0.062635 0.218143 0.006479

NC 0.000000 0.020134 0.087248 ... 0.046980 0.221477 0.013423

JJ 0.000000 0.006803 0.027211 ... 0.027211 0.482993 0.000000

PR 0.000000 0.010204 0.061224 ... 0.051020 0.275510 0.000000

CC 0.000000 0.000000 0.017857 ... 0.250000 0.196429 0.017857

PC 0.000000 0.085714 0.000000 ... 0.085714 0.171429 0.000000

CNC 0.000000 0.000000 0.000000 ... 0.000000 1.000000 0.000000

PRC 0.000000 0.000000 0.000000 ... 0.000000 0.375000 0.250000

VV 0.000000 0.000000 0.506289 ... 0.037736 0.103774 0.000000

ONC 0.000000 0.000000 0.000000 ... 0.250000 0.000000 0.000000

PP 0.000000 0.005291 0.015873 ... 0.089947 0.343915 0.010582

NN 0.001767 0.010601 0.113074 ... 0.051237 0.185512 0.005300

ON 0.000000 0.000000 0.055556 ... 0.111111 0.277778 0.111111

[17 rows x 17 columns]