Part of Speech Tagger for Hadiyyisa Language

Kacha Desta
HiLCoE, Computer Science Programme, Ethiopia
okoted@yahoo.com

Wondowssen Mulugeta
HiLCoE, Ethiopia
School of Information Science, Addis Ababa University, Ethiopia
wondwossen.mulugeta@aau.edu.et

Abstract

Part of Speech (POS) Tagging is the process of assigning the correct part-of-speech to each word in a text that best suits the definition of the word as well as the context of the sentence in which it is used. A part-of-speech tagger is used in many language technology applications as a first step in more complex NLP applications such as grammar checking, machine translation, information extraction and information retrieval. There are different approaches to the problem of POS Tagging. In this paper, we have used two approaches: Rule based and TnT approach. We have compared the performance of these techniques for tagging using combination with backoff taggers for Hadiyyisa Language. These approaches use supervised POS Tagging. It requires a large amount of annotated training corpus to build the tagging model and perform acceptable tagging of a new text. At the initial stage of POS Tagging of Hadiyyisa Language, we have very limited annotated corpus, i.e., 1280 manually tagged corpus of Hadyyisa sentences tagged with 32 identified tag sets and 10 basic tag sets. The 10 basic tag sets are derived from the 32 identified tag sets. The corpus was divided into 80% for training and 20% for testing. The research attempted to investigate a technique that maximizes the performance with this limited language resource. By experiment, the best configuration was investigated using TnT with Affix unknown word taggers, Rule-based with Affix initial state tagger, Rule based tagger when the initial state tagger with backoff in the sequence of TrigramTagger, BigramTagger, UnigramTagger, AffixTagger, and TnT-unknown word handling with backoff in the sequence of TrigramTagger, BigramTagger, UnigramTagger, AffixTagger. These approaches have accuracy of 66.64%, 64.65%, 72.54%, and 73.06% with 32 derived tag sets tagged corpus and 80.34%, 72.67%, 87.25%, and 89.03% with 10 basic tag sets tagged corpus respectively. Therefore, better performance is gained by the basic tag set tagged corpus with the tagger TnT-unknown word handling with backoff in the sequence of TrigramTagger, BigramTagger, UnigramTagger, AffixTagger taggers than the other approaches.

Keywords: Part-Of-Speech tagging; TnT; Rule based Tagger; Natural Language Processing

1. Introduction

Natural Language Processing (NLP) is a field of computer science, artificial intelligence and computational linguistics, which is concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human-computer interaction [1]. NLP is important in machine translation, fighting spam, information extraction, summarization, question answering, speech-recognition, etc.

There are different types of languages in the world where the basics in every language are words. Words are the building blocks of a language and word classes (Parts of Speech) in languages are the building blocks of sentences.

A word class (in other terms Part of Speech) is a set of words that display the same formal properties, especially their inflections, i.e., the differing forms one word takes to signal differing grammatical roles, e.g., boy-singular vs. boys-plural and distribution. Commonly recognized categories of parts of speech include: adverb, adjective, article, conjunction, noun, preposition, pronoun, and verb. There are other additional categories along the way, for example,

articles are generally taken to be a subset of a larger category called determiners [2].

Knowing all the parts of speech can help in advancing and understanding of the language because humans recognize patterns. Grammar is essentially a system of patterns applied to words that organizes them in a particular way. Without a grammatical system, we wouldn't be able to communicate with one another through reading, speaking, and writing [3].

POS tagging in particular is the act of assigning each word in a sentence to a tag that describes how that word is used in the sentence. That means POS tagging decides whether a given word used in a sentence is a noun, adjective, verb, etc. As Pla and Molina [4] noted, one of the most well-known disambiguation problems is POS tagging. A POS tagger attempts to assign the corresponding POS tag to each word in a sentence, taking into account the context in which this word appears [1].

In Ethiopia, part-of-speech tagger is developed for different languages and it is a demanding research area for other languages. Up to now, to the best of our knowledge, there are no attempts to investigate the suitable approaches of part of speech tagging for Hadiyyisa language.

Hadiyyisa is a language of the Hadiya people in Ethiopia. Most Hadiyyisa speakers live in the Southern Nations, Nationalities, and People's Region in the Hadiya Zone, the nearby zones and Special Woredas. Hadiyyisa uses Latin-based alphabets and it is an official language of Hadiya Zone and also it has been used as medium of instruction in primary schools, as a subject at junior secondary schools found in the zone and as field of study at Hossana Teachers Training College and Wachamo University and medium of transmission of local FM Radio programs.

Hadiyyisa has 27 letters including letters borrowed from other languages. Out of these, 23 are consonants phonemes and ten of them are both long and short consonants. Like other related languages, the basic word order in Hadiyyisa sentence is Subject Object Verb (SOV).

2. Related Work

In this section, different works on part of speech tagging in different languages are reviewed and they are selected based on their use of POS approaches and the language used.

Afaan Oromo part of speech tagger using HMM approach was developed by Getachew Mamo [7]. In this work, the researcher used HMM approach which is one of the most common mechanisms under stochastic approach. For training and testing purpose, 159 sentences were collected with a total of 1621 words to make the corpus balanced and used 17 tag sets.

In the tagging process, the tagger assigns word classes to a given Afaan Oromo text with two main phases. In the first phase, the tagger trains on the training data in order to compute and store both lexical and transitional probability of training data. In the second phase, the tagger accepts untagged Afaan Oromo text and is tokenized into words. Then the tagger assigns the correct POS tag for each token. This is achieved using unigram and bigram model of the Viterbi algorithm by taking the stored information during the first phase. The author tested the performance of the tagger using tenfold cross validation mechanism. As a result, 87.58% and 91.97% accuracy for unigram and bigram model respectively were achieved.

Another work on Part of speech tagging for Tigrigna using hybrid model was done by Teklay Gebregzabiher [6]. In this work a hybrid approach, which is the combination of rule based and HMM tagger, is designed. The author preferred the hybrid tagger because it performs better than the individual ones. A corpus with a total of 26, 000 words from different Tigrigna news broadcasting agencies was collected. The corpus prepared for this work was only news domain. Thirty six tag sets were identified and used in annotating the corpus for training the

HMM and rule based taggers. A supervised learning approach is used.

The author divided the corpus into training set and testing set for testing and training purposes. The training set consists of 75% of the corpus (20,000 words) and the testing set consists of 25% of the corpus (6,000 words). Different experiments are conducted for the three types of taggers namely the HMM tagger, the rule-based tagger and Hybrid tagger. Accordingly, 89.13%, 91.8%, and 95.88% performances are obtained for HMM, rule-based and hybrid taggers, respectively. It is concluded that the hybrid tagger for Tigrigna performs better than HMM tagger and rule-based tagger used independently.

3. The Proposed Solution

3.1 Data Collection and Corpus Preparation

A corpus can be defined as a systematic collection of naturally occurring texts, may be written or spoken [9]. For this work there is no readymade and balanced corpus developed before, so raw text was collected for corpus development from different datasets like Grade 5, 6 and 7 Hadiyyisa student textbooks, text transliterated from newsletters, University Hadiyyisa language teaching modules and Hadiyyisa Proverbs [10]. A total of 1280 sentences were collected and manually tagged with the 32 derived tag sets and another corpus is derived from the 32 tag sets tagged corpus to 10 basic tag sets tagged corpus.

3.2 Model Selection

We experimented on two kinds of POS taggers and compared their performance with different baseline taggers and combination with sequential backoff taggers in different sequences. The taggers used in this work are a transformational tagger (Brill algorithm) and a Hidden Markov Model (Tag 'n' Trigrams algorithm) approach.

TnT, the short form of *Trigrams 'n' Tags*, is a very efficient statistical part of speech tagger based on second order Markov models that is trainable on different languages and virtually any tag set. The component for parameter generation trains on tagged corpora. The system incorporates several methods of smoothing and handling unknown words [5].

The rule based approach, as its name indicates, relies on rules which are either handcrafted or machine learned. The rules are the important elements for annotating words in the rule based approach and require only small amount of training data and useful for limited domain.

3.3 Natural Language Toolkit and Python

The selected models are tested using the infrastructure of natural language processing which is NLTK using Python programming language [8].

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces and it has lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, semantic reasoning, and wrappers for industrial-strength NLP libraries [8]. It is an open source Python library for Natural Language Processing.

Python is a simple but powerful programming language with excellent functionality for processing linguistic data [8, 11]. It has efficient and high level data structure with simple but effective approach to object-oriented programming [8]. In this work Python 2.7.11 is used.

3.4 The Algorithms used with TnT and Brill Taggers

One class of sequential tagger is a regular expression. By using a regular expression, instead of looking for the exact word, we can define a regular expression that follows some patterns, and at the same time we can define the corresponding tag for the given expressions. For Hadiyyisa text, we developed the code shown below for some of the most common patterns to assign the different parts of speech. The designed regular expression helps to tag words using the TnT and Brill tagger as backoff tagger with other baseline taggers as unknown word handling and initial state tagger respectively. The

code below shows the Regular expiration for 32 derived tag sets for tagging the corpus.

```
RegexpTagger([
  (r'(.*o|.*oo|.*i|.*yo|.*ee|.*aa|.*
    kkoo|.*kkok)$', 'V'),
  (r'(.*nne|.*nnee)$', 'NPREP'),
  (r'(ee*|ka*)$', 'PROND'),
  (r'(an|An||I|i||Ki|ki)$', 'PRON'),
  (r'.*K$', 'ADJ'),
  (r'^[0-9]+(.[0-9]+)?$', 'CARDN'),
  (r'.*', 'N')])
```

The AffixTagger class is another ContextTagger subclass, but this time the context is either the prefix or the suffix of a word. This means the AffixTagger class is able to learn tags based on fixed-length substrings of the beginning or ending of a word. One can combine multiple affix taggers in a backoff chain if there is an interest to learn the tagging on multiple character length affixes [11]. We use the three suffix AffixTagger classes learning on -1, -2 and -3 character suffixes for Hadiyysa text based on the behavior of the language and used the tagger as base line combination as backoff tagger with other taggers:

```
from nltk.tag import AffixTagger
suf1_tagger = AffixTagger(dtrain,
  affix_length=-1)
suf2_tagger = AffixTagger(dtrain,
  affix_length=-2,
  backoff=suf1_tagger)
suf3_tagger = AffixTagger(dtrain,
  affix_length=-3,
  backoff=suf2_tagger)
backoff=suf3_tagger
```

3.5 Combining Taggers

A straight forward way to improve tagging is to combine different taggers sequentially, hoping to take advantage of the fact that different taggers are good at tagging different constructions. Combining classifiers in the context of POS tagging has mainly been used to increase tagging accuracy.

Backoff tagging is one of the core features of Sequential Backoff Tagger. It allows chaining taggers together so that if one tagger doesn't know how to tag a word, it can pass the word on to the next backoff tagger. If that one can't do it, it can pass the word on to the next backoff tagger, and so on until there are no backoff taggers left to check [8]. In this work, as a method to increasing performance we combined and evaluated Uni, Bi and Trigram taggers as sequential backoff tagging.

3.6 Evaluation

After developing the tagged corpus, even though our corpus is very small compared to the others like brown corpus in NLTK, the performance of TnT and Rule based taggers were evaluated using the corpus 80/20 percent for training and testing with different initial state tagger for Brill and different unknown word handling tagger for TnT tagger and combination of taggers by sequential backoff tagging in two different corpuses (i.e., tagged with the basic and the identified full tag set corpus). The reason behind considering basic tags is to determine how much the result differs between tests using basic data and tests making use of full data.

4. Results and Discussion

One of the main purposes of running tests of the taggers was to determine how accurate each tagging tool is, i.e., how much of the testing set input it tags correctly. Accuracy results for all taggers are presented below.

4.1 Results of Baseline Taggers

Before evaluating the performance of the TnT and Brill Tagger, we evaluated baseline taggers with their default options to see the performance on the two tag set tagged corpus and use for the combination (i.e., for initial and unknown word handling and combination as sequential backoff tagging) for the proposed taggers. Table 1 shows the results of the taggers on the two corpuses tagged with basic and derived tag sets.

Table 1: Baseline Tagger Performance

Corpus tagged with	*Default (%)*	*Affix (%)*	*Regex (%)*	*Unigram (%)*	*Bigram (%)*	*Trigram (%)*
Basic tag set	32.58	58.10	51.89	53.84	4.18	3.27
Derived tag set	31.85	50.51	33.45	49.56	3.40	2.71

As shown in Table 1, Affix Tagger performs better than the other taggers on the two corpuses when applied alone on Hadiyyisa corpus. But trigram and bigram taggers give lower results in accuracy whereas the Unigram and Regex taggers results are higher than Bi- and Tri-gram taggers.

The bigram and Trigram taggers manage to tag every word in a sentence during training, but does badly on an unseen sentence. As soon as it encounters a new word, it is unable to assign a tag. It cannot tag the following word even if it was seen during training, simply because it never saw it during training with none tag on the previous word. Consequently, the tagger fails to tag the rest of the sentence.

4.2 Results of TnT-HMM and Brill Tagger

To evaluate the performance of the taggers, 6 different experiments are conducted with different unknown word handling (smoothing) for TnT and initial state taggers for Brill taggers namely Default, Affix, Regular Expiration, Unigram, Bigram and Trigram tagger on two corpuses (i.e., basic and derived tag set tagged corpus). The default tagger assigns Noun (N) as a default word class based on frequency of individual tag within the training set. While in case of Affix we use -1, -2, and -3 suffix letters and unigram tagger assigns the most likely tag for a given word based on the frequency of individual tag within the training set. Table 2 shows the different experiments and their corresponding performance of the taggers. When the rule based tagger uses default tagger as initial state tagger, it assigns tags to every single word, even for words that have never been encountered before. From the result in Table 2, TnT with Affix unknown word handling tagger on basic tagset tagged corpus performs better than the others.

Table 2: TnT and Brill Tagger Performance

Tagger	*TnT Taggers with*		*Brill Taggers with*	
	Basic tag set Tagged Corpus %	*Derived tag set Tagged Corpus %*	*Basic tag set Tagged Corpus %*	*Derived tag set Tagged Corpus %*
Default	67.24	57.80	61.25	49.48
Affix	80.34	66.64	72.67	64.65
Regex	72.54	65.77	60.64	53.96
Unigram	54.01	50.39	53.92	49.69
Bigram	54.00	50.30	51.811	51.25
Trigram	54.01	50.37	50.99	50.38

4.3 Results of Combination Taggers with Backoff Tagging

There are different types of taggers used as the sequential backoff tagger such as Regexp Tagger, Affix Tagger, Unigram tagger, Bigram tagger and Trigram tagger. The backoff taggers are implemented in the Brill‘s and TnT taggers to detect and tag the word which is not tagged by the preceding tagger and the unknown words [8]. When unigram tagger assigns the most likely tag for a given word, the bigram and trigram taggers assign tags to words based on the preceding one and two words with their corresponding tags respectively [12].

Table 3: Combination of Taggers with Backoff Tagging Rule Based and TnT Tgagger

Tagger	*Accuracy %*	
Corpus tagged with	*Basic tag set*	*Derived tag set*
Brill-Initial state with Backoff Taggers		
• TrigramTagger, BigramTagger, UnigramTagger, RegexpTagger	72.37	65.25
• TrigramTagger, BigramTagger, UnigramTagger, AffixTagger	87.25	72.54
TnT -Unknown word Handling with Backoff Tagger		
• TrigramTagger, BigramTagger, UnigramTagger, RegexpTagger	72.54	65.78
• TrigramTagger, BigramTagger, UnigramTagger, AffixTagger	89.03	73.06

On the experiments from combinations of the above taggers (Regexp, Affix, Unigram, Bigram and Trigram taggers), comparatively good performance is gained by the TnT tagger with unknown word handling with backoff in the sequence of TrigramTagger, BigramTagger, UnigramTagger, AffixTagger and Rule based tagger when the initial state tagger with backoff in the sequence of TrigramTagger, BigramTagger, UnigramTagger, AffixTagger annotator respectively 73.06% and 72.54% on derived tag set and 87.25% and 89.03% on basic tag set tagged corpus.

5. Conclusion and Future Work

Part of speech tagging is a very important step in most advanced language technology systems. It is a nontrivial problem due to ambiguous words and unknown words, i.e., words are not in the lexicon. POS tagging is harder for some languages than others. A straight forward way to improve tagging is to combine different taggers, hoping to take advantage of the fact that different taggers are good at tagging different constructions. This work selected a Rule based, TnT and combination by backoff chain taggers at sentence level for Hadiyyisa language.

Corpora are important for many types of linguistic research including part of speech tagging. We used a corpus with a total of 1280 sentences which is collected from different genres. There are several standard tag sets used in corpora and in POS tagging experiments. This work used 32 part of speech tags which are identified as a tag set for annotating a raw text and reduced version with 10 basic tag sets and used two different corpuses which are tagged with 10 and 32 tag sets.

The tagged corpus is divided into training and testing set. The training set comprises of 80% of the corpus and the reaming 20% of the corpus is used as a testing set.

In this work NLTK and Python are used in the experimentation of Hadiyyisa part of speech taggers. To test the performance of the taggers different experiments are conducted for the two types of taggers namely the Rule-based and TnT and also combination with backoff on the two different (basic and identified derived tag set) tagged corpus. As a result, 64.65%, 66.64%, 72.54% and 73.06% performances are obtained for Rule-based with Affix Tagger initial state tagger, TnT with Affix unknown word taggers, and Rule based tagger when the initial state tagger is backoff in the sequence of TrigramTagger, BigramTagger, UnigramTagger, AffixTagger, and TnT-unknown word handling with backoff in the sequence of TrigramTagger, BigramTagger, UnigramTagger, AffixTagger, respectively on identified 32 tag sets and 80.34%, 72.67%, 87.25% and 89.03% results were obtained for the basic 10 tag set tagged corpora respectively. From the experiment the performance using basic tag set tagged corpus performs better than the full identified tag set tagged corpus and also higher performance is gained by combining taggers for

initial state tagging and unknown word smoothing at sentence level for Brill and TnT taggers respectively.

For the future, we would like to extend the work in the following areas.

- Preparation of a balanced corpus that contains texts which represent different genres like Bible, Newspapers, Fiction, Textbooks, Parliamentary Reports, etc.
- Comparative study of different approaches HMM based, Rule-based, and Artificial Neural Network based taggers for Hadiyyisa language with more training and testing data.

References

[1] Daniel Jurafsky and James Martin, "Speech and Language Processing: An Introduction to Natural Speech Recognition", Prentice-Hall, Inc, New Jersey, 2000.

[2] Kim Ballard, "The Frameworks of English: Introducing Language Structures", 3rd ed. Palgrave Macmillan, 2013.

[3] Simon and Schuster, "Handbook for Writers; Psychology Encyclopaedia: Pattern Recognition", Lynn Quitman Troyka, 2002,

[4] Ferran Pla and Antonio Molina, "Natural Language Engineering: Improving Part of-Speech Tagging using Lexicalized HMMs", Cambridge University Press, UK, 2004.

[5] Brants, Thorsten, "TnT – A Statistical Part-of-Speech Tagger", In Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP 2000), Seattle, WA, 2000.

[6] Teklay Gebregzabiher, "Part of Speech Tagger for Tigrigna Language" Unpublished Masters Thesis, Addis Ababa University, 2010.

[7] Getachew Mamo, "Part-of-Speech Tagging for Afaan Oromo Language", Unpublished Masters Thesis, Addis Ababa University, 2009.

[8] Steven Bird, Ewan Klein, and Edward Loper. "Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit", O'Reilly Media, 1st ed, Cambridge, 2009.

[9] Nadja Nesselhauf, "Corpus Linguistics: A Practical Introduction", 2011.

[10] Getahun Waaxummo, "Hadiyy Heessechchaa Kobillishsha", Nigd Mattemiy Dirijjit, undated.

[11] Python Programming Language: http://www. python.org, Last Accessed on Aug. 8, 2013.

[12] Jurafky. D. and Martin. H. James, "Speech and Language Processing, An introduction to Natural Language Processing, Computational Linguistics and speech recognition", 2nd ed. New Jersey, Prentice Hall, 2006.