Global Scientific JOURNALS

GSJ: Volume 8, Issue 6, June 2020, Online: ISSN 2320-9186
www.globalscientificjournal.com

Supervised morphosyntactic tagging of parts of speech of Twi, a Ghanaian language

Joseph Arimiyawu, Abdulai
Affiliation : Odomaseman Senior High School, Odomase, Sunyani, Ghana.
Email : abdulaijoseph@ymail.com

Richard, Okyere Baffour
Affiliation : University of Energy and Natural Resources, Fiapre, Ghana
Email : richard.okyere@uenr.edu.gh

Abstract

In this article, we present the results of the supervised automatic tagging of parts of speech of Twi. We speak of the importance of tagging parts of speech as presented by other researchers. We explain the objective of the present work and how tagging the parts of speech of the Twi language is useful. We present the corpus as well as the tagging tool which we adapted for the Twi language. We also present the methodology and the steps involved in tagging. We analyse some morphosyntactic phenomena which can be a source of difficulty to the automatic tagging process. We suggest some solutions to these problems. In conclusion, we present some recommendations aimed at improving the results of this preliminary approach to the automatic tagging of the Twi language.

Keywords

Twi, part of speech tagging, treetagger, Natural Language Processing (NLP)

1. Introduction

Morphological tagging is a process that involves assigning a tag to each word in a text. This is important since the information that is provided for each word and its surroundings is necessary for linguistic analyses. In the field of Natural Language Processing (NLP), morphological tagging is used for speech synthesis, linguistic searches based on corpora, and translation [7]. According to a study on the development of morphological tags for Arabic [11], providing text with linguistic information (morphological tags) increases the potential of the text to be integrated into various computer applications for linguistic analysis.

Twi is one of the most widely spoken languages in Ghana. The Akan group is made up of several languages, including Twi, which was the subject of our study. According to the classification carried out by [6], the Akan belongs to the kwa branch of the big Niger-Congo family. According to [1] the other languages of the Akan group are: *fante, ahanta, aowin, sefwi, bono, ahafo, kwahu, akyem, agona, dankyira and asen.*

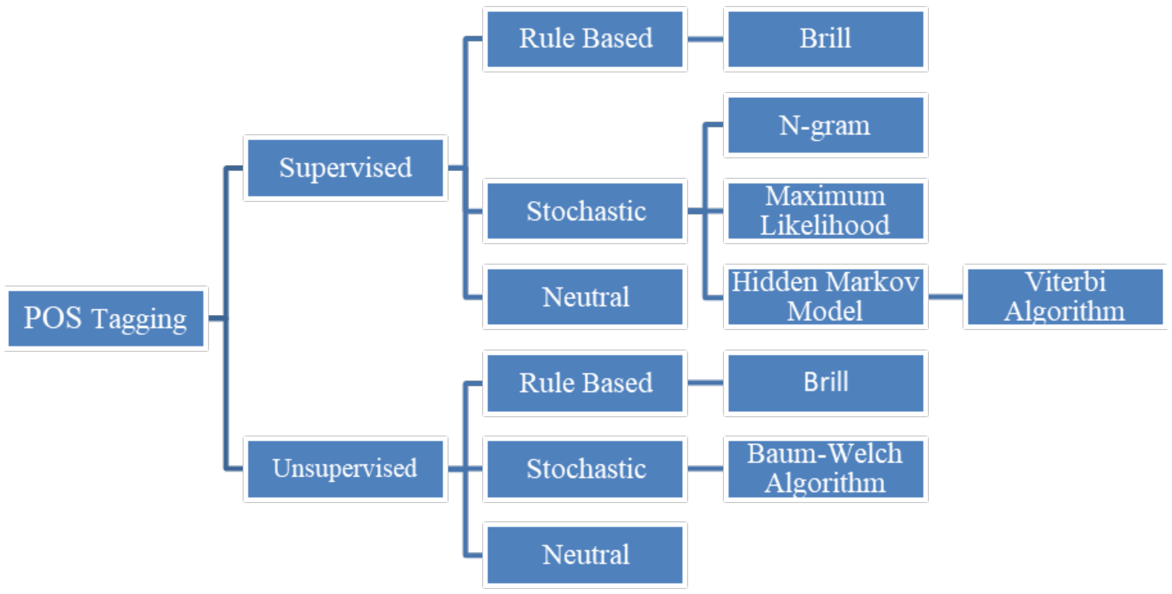
According to research, there are two versions of the grammar of Twi. First, there is the grammar proposed by [4] and the modified version of [2]. According to these two versions, there are nine parts of the speech for the Twi language: *Edin (The noun), Edin Nkyerɛkyerɛmu (The adjective), Edinnsiananmu (The pronoun), Adeyɔ (The verb), Ɔkyerɛfoɔ (The adverb), Edin -akyi sibea (Postposition), Nkabomdeɛ (The conjunction / connector), Nteamu (Interjection), Nsisodeɛ (The emphasis marker).* The present work was carried out on these nine parts of the speech.

In this article, we first present the literature review. Next, we present the methodology used and the corpus of the study. We describe the tool we used. We also present the pre-treatment of the corpus and the steps we followed for tagging. Finally we present the results, a discussion of the results and perspectives for future research.

2. Literature review

Over time, automatic morphological tagging has undergone a lot of development, which has led to the development of several tagging methods as well as tools that apply these methods. We present in the figure below some tagging methods [7].

Figure 1 : Classification of methods of tagging.



We find that the supervised and unsupervised tagging methods share three components: the use of rules, the stochastic method and the neutral method. The difference between the two tagging methods is marked by the use of a set of predefined rules and a training corpus (supervised method) or the use of a set of predefined rules, the context of use of words without a training corpus (unsupervised method). An example of the rules used in this context could be as follows: a word preceded by a determinant and followed by an adjective should be a noun [7].

Regarding the stochastic method, we determine the tags to assign to words by calculating the probability that a word is associated with a certain tag and also, the frequency of such an association. This probabilistic method is used in the *TreeTagger*. We also have tools such as Brill's tagger [3] which uses the two components mentioned above (rules and probability calculation). This tool works well for languages which do not have a sufficient corpus for analysis but which have a well-established rule system. Besides the Brill tagger, other taggers have been tested on several languages [11].

2.1. TreeTagger

The *TreeTagger* is a supervised probabilistic tagging tool that works according to decision trees. This tool is based on the principle of "Hidden Markov Model", a representation model of the distribution of probabilities in relation to a series of observations [5]. Designed by

Helmut for English [8], this tagger has been trained and adapted to German [9] and other languages such as French, Italian, Dutch, Spanish, Bulgarian, Russian, Portuguese, Galician, Chinese, Swahili, Slovak, Latin, Estonian, Polish, and Old French. The tagger is supposed to be able to label other languages apart from those mentioned above if these languages have a lexicon and a manually tagged training corpus. To tag a language with *TreeTagger*, a training model is created from a sample of the corpus. The creation of the training model is ensured by the “train-tree-tagger” module which is launched at the command line. This training module requires four arguments:

1. “Lexicon”: a lexicon composed of words. On each line of the lexicon, there is a word and its lemma separated by a tabulation.

2. "open class file": a file containing the labels that are used when the tagger is dealing with unknown words.

3. "input file": this is the file that contains the manually tagged corpus. This file consists of a word and its appropriate label on each line.

4. “output file”: this is the name of the file where the training results are stored.

Following the creation of this model, the tagger is launched with another untagged sample.
The module that provides automatic tagging requires three arguments:

1. "parameter file": the file created at the end of the training phase (this is the "output file" of the previous steps).

2. “input file”: this file contains the text to be automatically tagged. There is a word on each line of the file.

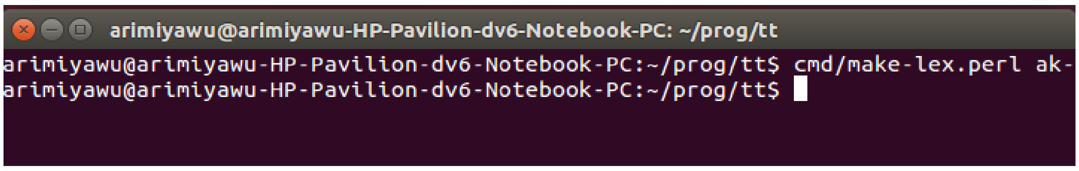
3. "output file": the results of the automatic tagging are stored in this output file.

3. Methodology

The first step is to process the corpus to make it readable to the tagging software. As part of tagging with *TreeTagger*, we manually tagged a sample of 1000 words from our corpus to train *TreeTagger*. This training took place over several phases. In each phase of the training, we added 100 tokens to the training corpus. Then we automatically tagged another 100 words from the untagged corpus sample with *TreeTagger*. Following the automatic tagging, we checked the labels and made corrections for the lexical units which were incorrectly tagged by the tool. After verification, we introduced a new sample of 100 tokens to the training corpus and to the corpus to be tagged automatically. We repeated this cycle until the 1000 tokens sample was properly tagged. Then, we proceeded to the last phase of tagging the entire corpus of 36,000 tokens.

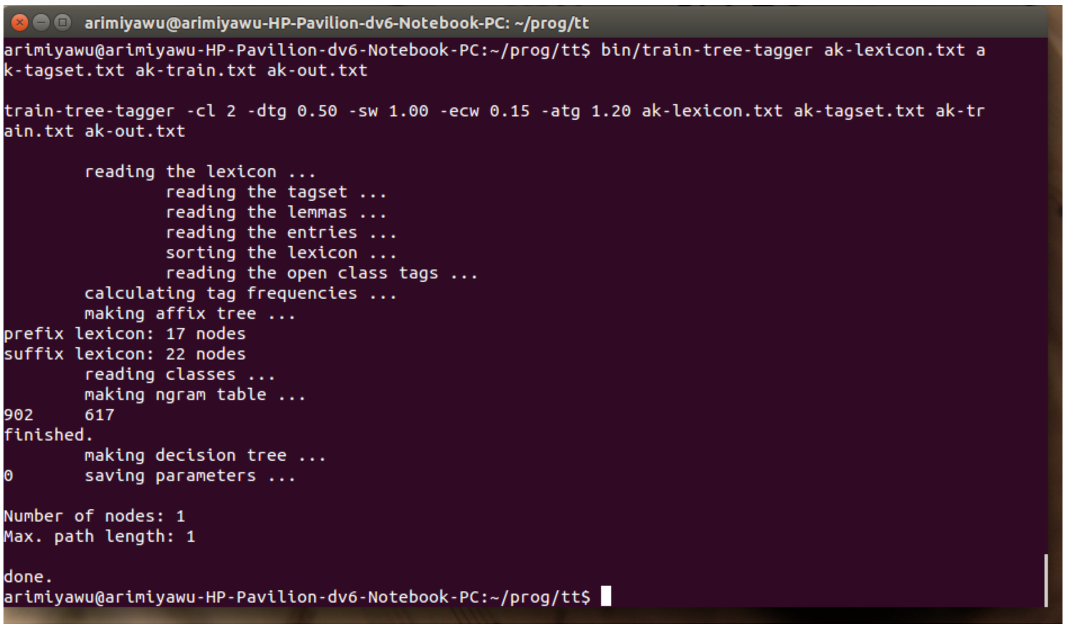
For the tagging process, we needed three main commands which we executed at the command line. The first command (Figure 2 below) that we executed was used to create a lexicon from the well-tagged training corpus. This command took as argument the file of the training corpus and the file in which the results were saved. The second command (Figure 3 below) was the one used to create the parameters that were used for automatic tagging. This command took four arguments: the lexicon (created after execution of the first command), the file of predefined tags (a set of morphological and syntactic tags), the training corpus and finally the output file where the parameters have been saved. The third command (Figure 4 below) was used for automatic tagging. This command took three arguments: the parameters file (created after execution of the second command), the file with tokens without labels and the output file where the results of the automatic tagging were saved. The figures below demonstrate the operation of the three processes.

Figure 2 : Creation of the lexicon.



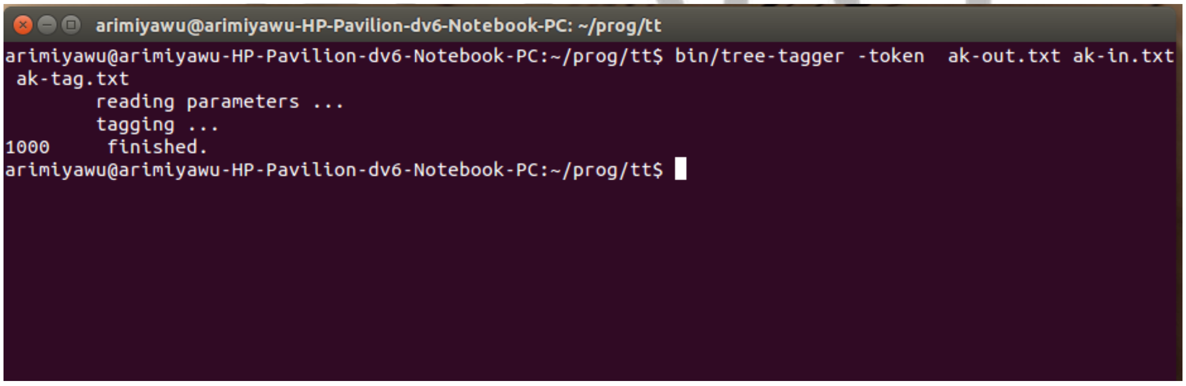
```
arimiyawu@arimiyawu-HP-Pavilion-dv6-Notebook-PC:~/prog/tt$ cmd/make-lex.perl ak-
arimiyawu@arimiyawu-HP-Pavilion-dv6-Notebook-PC:~/prog/tt$
```

Figure 3 : Training phase of TreeTagger.



```
arimiyawu@arimiyawu-HP-Pavilion-dv6-Notebook-PC: ~/prog/tt
arimiyawu@arimiyawu-HP-Pavilion-dv6-Notebook-PC:~/prog/tt$ bin/train-tree-tagger ak-lexicon.txt a
k-tagset.txt ak-train.txt ak-out.txt

train-tree-tagger -cl 2 -dtg 0.50 -sw 1.00 -ecw 0.15 -atg 1.20 ak-lexicon.txt ak-tagset.txt ak-tr
ain.txt ak-out.txt

        reading the lexicon ...
                reading the tagset ...
                reading the lemmas ...
                reading the entries ...
                sorting the lexicon ...
                reading the open class tags ...
        calculating tag frequencies ...
        making affix tree ...
prefix lexicon: 17 nodes
suffix lexicon: 22 nodes
        reading classes ...
        making ngram table ...
902     617
finished.
        making decision tree ...
0       saving parameters ...

Number of nodes: 1
Max. path length: 1

done.
arimiyawu@arimiyawu-HP-Pavilion-dv6-Notebook-PC:~/prog/tt$
```

Figure 4 : Automatic tagging with TreeTagger.



```
arimiyawu@arimiyawu-HP-Pavilion-dv6-Notebook-PC: ~/prog/tt
arimiyawu@arimiyawu-HP-Pavilion-dv6-Notebook-PC:~/prog/tt$ bin/tree-tagger -token  ak-out.txt ak-in.txt
 ak-tag.txt
        reading parameters ...
        tagging ...
1000     finished.
arimiyawu@arimiyawu-HP-Pavilion-dv6-Notebook-PC:~/prog/tt$
```

4. Presentation of the corpus

We worked with a corpus of 36,000 words collected using the web crawler “An Crúbadán” [12]. This tool was designed for the creation of corpora for under resourced languages . Three main sites were used for the creation of the corpus namely: Wikipedia (5%), the Jehovah's Witnesses / Bible website (40%) and the Universal Declaration of Human Rights website (5%). In addition, there were stories (5%), song lyrics (10%) and a few news articles written in Twi (35%).

The Jehovah's Witnesses / Bible site made up the bulk of the corpus given the volume of publications and translations of religious documents by this organization. After that, there were news articles which contained information from several fields including politics, law, transportation, business. In recent years, technological advancement has made it easier to transcribe song lyrics even for languages like Twi. This is why we see some traces of the lyrics of the most popular songs in twi. A little Google search with “twi ghanaian language” for example showed us some Wikipedia pages where we found some information written in twi: how to say “yes” or “no”, “good night”, “how are you you? ”,“ excuse me ”,“ what is your name? ”,“ I'm going ”.

By analysing the corpus closely, we found that it was composed of at least five elements:

1. Words belonging to the lexicon of twi
2. Loan words from the English language (for most occurrences)
3. Punctuation marks (, /. /? /: /; /! / - /)
4. Special characters such as parentheses () / [] / {}, quotes (“” ‘’), ©, *,%,
5. Numbers (dates)

The corpus was stored in a file (.txt) because this format is the most common and best suited for automatic processing by the computer tool that we used for tagging.

4.1. Pre-treatment of the corpus

Morphological tagging is carried out on the lexical words of a language. Before tagging a corpus, it is necessary that this corpus is adapted to a format which will be accepted and processed by the taggers. The stages of adaptation of the corpus are governed by the morphological properties of the language: the delimitation of words, affixation. An inflectional language like Arabic requires rigorous pre-processing steps in order to provide the tagger with analysable data.

One study that presents a morphological tagger for Arabic lists five pre-treatment steps and a sixth step where the real job of tagging is done [11]. First, a segmentation module included in the NLTK (Natural Language ToolKit) software segments the text to have words, punctuation marks, symbols (for example currency), numbers, Latin characters and HTML (Hypertext Markup Langauge) or XML (Extensible Markup Language) labels. Then, another module divides the words into three parts: the proclitic and the prefix; the root ; and finally the suffix and the enclitic. Following this second step comes a third step of checking the roots in a lexicon. Then in the fourth step, the words found during the previous steps are associated with a diagram appearing in a list of 2,730 verbal diagrams or 985 nominal diagrams supplied beforehand to the tagger. These diagrams already have predefined tags. The fifth step is to

relate the short vowels of the predefined patterns to the words that are associated with the patterns. Finally, a general tag is assigned to the word, taking into account all the morphemes that combine to form this word. The word w*asayakatabaūnahā* for example will be divided into morphemes *wa sa ya kataba ūn hā*. A tag will be assigned to each part and then the tags are combined and assigned to the word.
The corpus that we used for tagging has also been pre-processed in order to adapt it to the format that was parsable by the *TreeTagger*. Using a python script, we segmented the text of the corpus to have tokens (a word on each line of the file). This format was required by the software used (*TreeTagger*). The segmentation of the words was done in relation to the spaces; that is, a lexical word is an element that is between two spaces. The file we had at the end of this pre-processing phase served as a basis for the other stages of the tagging procedure (training and automatic tagging).

5. Results

In the table that we present below, "TreeTag" represents the tags that were automatically assigned to the elements to be tagged. In cases where the "TreeTag" tag was not precise, we proposed a good tag represented by "GoodTag". We launched the TreeTagger for the first time on a sample of 100 untagged items and the results of this step show that the most common label was "Nc". As a result, this tag was automatically assigned to most unknown words. At this point, our training corpus was made up of 100 hand-tagged elements. The results of this first phase of automatic tagging are presented in the table below.

Table 1 : First phase of automatic tagging.

Token	Treetag	Good tag
Wobetumi	Nc	2PSFUTV
afrɛ	Nc	PerfV
!	Nc	SENT
Bi	Nc	INDEF
Hena	Nc	Pin
wei	Nc	PrDem
wɔ	POST	V
905-890-1010	1PP	Card
Anaa	Nc	Concor
1-800-668-1146	PrDem	Card
,	SENT	

The table above shows some of the results we had after the first phase of automatic tagging. All personal pronouns detached from verbs were properly tagged. Personal pronouns that had bad labels were those that were attached to verbs. The verbs that appeared alone in this sample of 100 untagged items were properly tagged with the label "V". On the other hand, verbs which were attached to personal pronouns (constructions of the *personal pronoun -verb* type) and constructions composed of a personal pronoun, a time indicator and a verb (all glued together) were not properly tagged. For this last type, we noted four constructions on the set of 100 elements without labels: *wɔ-bɛ-ma, wɔ-a-we, wɔ-bɛ-kyerɛ. Wo-bɛ-tumi.* For the second phase of automatic tagging, we proposed composite labels for the constructions that we have just mentioned.

Table 2 : Second phase of automatic tagging.

Token	TreeTag	Good tag
Ahmadiyyat	NP	
Soro	Nc	
Ne	Concor	
Kristo	Nc	
Akodi	Nc	PerfVV
Ade	Nc	
Luka	Nc	NP
12:32	Card	
;	SENT	
Adi	Nc	
.	SENT	
Yesu	NP	
Ne	Concor	
Nkwa	Nc	
No	DEF	
–	Nc	SENT
Wia	Nc	
Yi	Adj-Dem	
Ase	PerfV	

At the second phase of training and automatic tagging, the size of the training corpus was increased to 200 well-tagged elements. Another sample of 200 untagged items was also added to the input corpus to test the effectiveness of the tagger. We found that the label "Nc" was always displayed in cases where the tagger encountered an unknown word. However, most of the tokens that carried this label were properly tagged, that is, they were common nouns. The numbers which were incorrectly tagged at the end of the first phase of automatic tagging, were properly tagged after this second phase. The tag “Card” (cardinal numeral adjective) was assigned to elements composed of numbers as we see in Table 2 above. This second phase gave us an improved basis for the third phase of training and automatic tagging.

Table 3 : Third phase of automatic tagging.

Token	Treetag	Good tag
wɔn	3PP	
Ankasa	Nc	EMP
Adwuma	Nc	
.	SENT	
Wɔn	Nc	3PP
yɛ	V	
nyamesurofoɔ	Nc	
papaafoɔ	Nc	
Nyinaa	Adv	
Te	V	
Sei	Nc	
,	SENT	
Na	Concor	
wɔn	3PP	
Ndi	Nc	
Abronsam	Nc	
Som	Nc	
Akyi	Nc	
sɛ	PRel	
deɛ	Nc	
w'ate	Nc	
No	DEF	
.	SENT	
Sarkodie	Nc	NP
Medwene	Nc	1PSV
sɛ	PRel	
m'ano	Nc	
Ate	Nc	PerfV

At the third phase of automatic tagging, the training corpus consisted of 600 well-tagged elements. We arrived at this point by modifying the results of the previous phases and adding them to the training corpus. Until then, we found that the more we increased the size of the training corpus, the more the results of the automatic tagging improved as we can see in Table 3 above.

Table 4 : Fourth phase of automatic tagging.

Token	Treetag	Good tag
De	V	
Wo	2PS	
Ho	POST	
bɔ	Nc	V
deɛ	Pin	
Aka	Nc	
No	DEF	
A	EMP	
,	SENT	
ɛyɛ	3PSV	
yɛ	V	
,	SENT	
ɛyɛ	3PSV	
deɛ	Pin	
Wo	2PS	
pɛ	V	
.	SENT	
Baanodifoɔ	Nc	
nsusuyɛ	Nc	
Baanodifoɔ	Nc	
De	V	
wɔn	3PP	
nsusuyɛ	Nc	
A	EMP	
ɛfa	3PSV	
Sukuu	Nc	
No	DEF	
Ho	POST	
bɛma	Nc	FUTV
Sukuu	Nc	
Panyin	Nc	
.	SENT	
2	Card	
.	SENT	
Obiara	Nc	

After the final training and automatic tagging phase, there was a total of 1000 items properly tagged in the training corpus. This well-tagged final sample of 1000 words was used for automatic tagging of the entire corpus. By varying the sample of the untagged corpus each time we did the automatic tagging, we exposed the tagger to the maximum of new words in order to increase the precision when tagging the entire corpus.

6. Discussion

It is obvious that the results were good when we worked with a fairly large training corpus and when the text to be tagged was not too large. The TreeTagger which we used for our work was first designed for English and then German and other languages. For these languages, the algorithms have been developed in such a way that they are able to label unknown words by making guesses from the words which were well tagged during the training phase.

6.1. Some linguistic phenomena

One of the problems we faced was the problem of segmentation. This same problem was reported by other authors who worked on tagging languages that behave like Twi when it comes to word form and composition. According to a study on the tagging of Urdu [10] the problem of suffixing in the composition of words is mentioned. According to this study, words are delimited by spaces. However, suffixes (which are not words) are considered lexical words since there is a space between them and their radicals. This makes it difficult for taggers to automatically assign tags. Regarding our work, we noted a problem of segmentation linked to the attachment of personal pronouns to verbs, the time indicators stuck to verbs and also with serial verb constructions.

Table 5 : Segmentation of personal pronoun-verb constructions.

Token	Tag
Ɔyɛ	3PSV
Ɔsom	3PSV
Yɛhyɛ	1PPV
Medwene	1PSV

In the table above, we find some examples of constructions where the personal pronoun is stuck to verbs. We treated them as non-separable syntactic phenomena. This is why we also formulated special tags to represent the two units of the construction.

Table 6 : Time markers of verbs.

Token	Tag
Wo-bɛ-tumi	2PS-FUT-V
Wo-bɛ-kyerɛ	2PS-FUT-V
mɛ-yɛ	1PS-FUT-V
ɛ-bɛ-ma	3PS-FUT-V
Ɔ-bɛ-hwɛ	3PS-FUT-V
A-yɛ	Perf-V
A-si	Perf-V
A-tumi	Perf-V
A-tena	Perf-V
A-kɔ	Perf-V
wɔ-a-we	3PP-Perf-V
ɛ-re-ba	3PS-Prog-V
A-ko-di	Perf-V-V

In the table above, we present some examples of verbs that have time markers. Here, we see three time markers: *bɛ*, *a* and *re* for the future, the perfect aspect and the progressive aspect respectively. The labels we provide for these constructions are also a combination of the time markers and the verbs.

7. Perspective for future research

As part of our work, we trained the tagger with a training corpus; that is, the performance of the tagger depended on the quality of the data we provided in the training corpus. We believe that it is necessary to modify the algorithms so that they can determine the word tags, taking into account the conditions under which these words appear. For example: a word preceded by a time marker such as *bɛ* is a verb. By defining these parameters, the tagger will be able to determine certain tags independently without us really having to supply them in a training corpus. We plan to find a larger corpus to increase the size of the training corpus and test the effectiveness of the parameters we have defined for tagging the Twi language.

8. Conclusion

In this study, we presented an approach to tagging the parts of speech of Twi. We worked with a corpus of 36,000 words. We used a tagger (TreeTagger) to test its effectiveness by analysing the results obtained. The tagging took place over three training phases, at the end of which the automatic tagging of the entire corpus was done. Our work has contributed to the enrichment of studies that focus on Twi since we shed light on fundamental phenomena that have been the subject of previous studies. Also, this work is a first step in the production of a database that will be integrated into linguistic tools for the automatic processing of the Twi language. This database is composed of lexical words endowed with morphological information by means of the automatic tagging of the Twi that we carried out within the framework of this work.

Acknowledgements

We would like to thank Mr. Scannell who provided us with the corpus on the Twi language. This corpus formed the basis of our work. We also thank all those who helped us financially.

References

[1] Adu Manyah, K. (2002) *Introduction à la phonétique et à la phonologie africaines: les sons de tous les jours*. Paris, France : L'Harmattan.

[2] Agyekum, K. (2010) *Akan kasa nhyehyɛeɛ*. Accra : Dwumfuor Publications.

[3] Brill, E. (1992) A simple rule-based part of speech tagger. *Proceedings of the workshop on Speech and Natural Language*, 112–116. http://dl.acm.org/citation.cfm?id=1075553

[4] Christaller, J. G., (1964) A grammar of the Asante and Fante language called Tshi Chwee, Twi : based on the Akuapem dialect with reference to the other Akan and Fante dialects repr. of 1875 . Farnborough, Hants.: Gregg Press.

[5] Ghahramani, Z. (2001) An Introduction to Hidden Markov Model and Bayesian Networks. *International Journal of Pattern Recognition and Artifical Intelligence*. 15(1), 9-42.

[6] Greenberg, J. H. (1963) *The Languages of Africa*. International journal of American linguistics, 29(1).

[7] Hasan, F. M., Uzzaman, N., & Khan, M. (2007) Comparison of different POS Tagging Techniques (N-Gram, HMM and Brill's tagger) for Bangla. *Advances and Innovations in Systems, Computing Sciences and Software Engineering*, 121–126. http://link.springer.com/chapter/10.1007/978-1-4020-6264-3_23

[8] Helmut S. (1994) Probabilistic Part-of-Speech Tagging Using Decision Trees. *Proceedings of International Conference on New Methods in Language Processing*. http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger1.pdf

[9] Helmut S. (1995) Improvements in Part-of-Speech Tagging with an Application to German. *Proceedings of the ACL SIGDAT-Workshop*. http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger2.pdf

[10] Humera Khanam, M., Madhumurthy, K.V. & Khudhus, M.D.A. (2013) Comparison of TnT, Max. Ent, CRF Taggers for Urdu Language. *International Journal of Engineering Sciences Research-IJESR*. 4(1), 1164-1168.

[11] Sawalha, M., & Atwell, E. S. (2010) Fine-grain morphological analyzer and part-of-speech tagger for Arabic text. *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)* 1258–1265. http://eprints.whiterose.ac.uk/42641/

[12] Scannell, K. P. (2007) The Crúbadán Project: Corpus building for under resourced languages. Building and Exploring Web Corpora. *Proceedings of the 3rd Web as Corpus Workshop*, 5–15.

Annexe 1

DEF	Definite Article
INDEF	Indefinite Article
NAbs	Abstract Nom
PRel	Relative Pronoun
PrDEM	Demonstrative Pronoun
PrINDEF	Indefinite Pronoun
PrPers	Personal Pronoun
Pin	Interrogative Pronoun
P	Pronoun
Nc	Common Noun
N	Noun
NP	Proper Nom
PC	Past Tense
IMP	Imperfect
Imp	Imperfect Aspect
Perf	Perfect Aspect
FUTP	Near Future
FUTL	Distant Future
ADJ-DEM	Demonstrative Adjective
ADJPO	Possessive Adjective
ADJ-Int	Interrogative Adjective
AQA	Qualitative Adjective
AQE	Qualitative Adjective
Pr-PO	Possessive Pronoun
EMP	Emphasis Marker
NG	Negation
PAT	Particle
ME	Emphasis Marker
AdvL	Adverb of place
AdvT	Adverb of time
AdvM	Adverb of Manner
AdvD	Adverb of Degree

AdvF	Adverb of Frequency
ANC	Cardinal Numeral Adjective
Card	Cardinal Adjective
ANO	Ordinal Adjective
ConCor	Conjunction of coordination
ConSub	Conjunction of subordination
Pper	Personal Pronoun
1PS	First person singular
2PS	Second person singular
3PS	Third person singular
1PP	First person plural
2PP	Second person plural
3PP	Third person plural
V	Verb
Imper	Imperative
Int	Interjection
Con	Connector
Prog	Progressive
Pr	Present
Sg	Singuler
Pl	Plural

Annexe 2

Token	TreeTag	Good Tag
w'ayi	Nc	
wɔn	3PP	
Adi	PerfV	
pɛn	Nc	EMP
No	DEF	
Ahyɛase	Nc	
Foforo	Nc	Adv
;	SENT	
Ne	Concor	
yɛn	1PP	
Botae	Nc	
sɛ	PRel	
yɛbɛboa	Nc	1PPFUTV
Sukuufo	Nc	
Ama	PerfV	
wɔn	3PP	
Aduru	Nc	
wɔn	3PP	
Botae	Nc	
Ho	POST	
.	SENT	
Woa	Nc	2PS
hwɛ	Nc	V
Family	Nc	
Guy	Nc	
,	SENT	
dieɛ	Nc	
Brian	Nc	
De	V	
Stewie	Nc	
Ka	Nc	
No	DEF	
?	SENT	
Sɛ	PRel	
yɛyɛ	Nc	1PPV
Obi	Nc	PrDem
bɔne	Nc	Adj
A	EMP	
,	SENT	
yɛsrɛ	Nc	1PPV
bɔne	Nc	Adj
fakyɛ	Nc	VV

Na	Concor	
Ama	PerfV	
yɛne	Nc	
Onii	Nc	PrDem
korɔ	Nc	
No	DEF	
Ntam	Nc	
Adwo	Nc	PerfV
.	SENT	
(	SENT	
1	Card	
)	SENT	
Dɛn	Pin	
Nti	EMP	
Na	Concor	
wɔfrɛ	Nc	
No	DEF	
“Asɛm	Nc	
no”	Nc	
?	SENT	
Sɛ	PRel	
wopɛ	2PSV	
sɛ	PRel	
Wo	2PS	
De	V	
Wo	2PS	
Ho	POST	
bɔ	Nc	V
deɛ	Pin	
Aka	Nc	
No	DEF	
A	EMP	
,	SENT	
ɛyɛ	3PSV	
yɛ	V	
,	SENT	
ɛyɛ	3PSV	
deɛ	Pin	
Wo	2PS	
pɛ	V	
.	SENT	
Baanodifoɔ	Nc	
nsusuyɛ	Nc	
Baanodifoɔ	Nc	
De	V	
wɔn	3PP	
nsusuyɛ	Nc	

A	EMP	
ɛfa	3PSV	
Sukuu	Nc	
No	DEF	
Ho	POST	
bɛma	Nc	FUTV
Sukuu	Nc	
Panyin	Nc	
.	SENT	
2	Card	
.	SENT	
Obiara	Nc	