Towards POS Tagging Methods for Bengali Language: A Comparative Analysis

Chapter · February 2021

DOI: 10.1007/978-3-030-68154-8_93

Fatima Jahara
Workera.ai
4 PUBLICATIONS **30** CITATIONS
SEE PROFILE

Adrita Barua
Chittagong University of Engineering & Technology
2 PUBLICATIONS **19** CITATIONS
SEE PROFILE

Asif Iqbal
Chittagong University of Engineering & Technology
12 PUBLICATIONS **88** CITATIONS
SEE PROFILE

Avishek Das
Chittagong University of Engineering & Technology
30 PUBLICATIONS **99** CITATIONS
SEE PROFILE

Towards POS Tagging Methods for Bengali Language: A Comparative Analysis

Fatima Jahara, Adrita Barua, MD. Asif Iqbal, Avishek Das, Omar Sharif , Mohammed Moshiul Hoque* , and Iqbal H. Sarker

Department of Computer Science & Engineering, Chittagong University of Engineering & Technology, Chittagong-4349 Bangladesh
fatimajahara@ieee.org, {adrita766, asifiqbalsagor123, avishek.das.ayan}@gmail.com, {omar.sharif, moshiul_240*,iqbal}@cuet.ac.bd

Abstract Part of Speech (POS) tagging is recognized as a significant research problem in the field of Natural Language Processing (NLP). It has considerable importance in several NLP technologies. However, developing an efficient POS tagger is a challenging task for resource-scarce languages like Bengali. This paper presents an empirical investigation of various POS tagging techniques concerning the Bengali language. An extensively annotated corpus of around 7390 sentences has been used for 16 POS tagging techniques, including eight stochastic based methods and eight transformation-based methods. The stochastic methods are unigram, bi-gram, tri-gram, unigram+bigram, unigram+bigram+trigram, Hidden Markov Model (HMM), Conditional Random Forest (CRF), Trigrams 'n' Tags (TnT) whereas the transformation methods are Brill with the combination of previously mentioned stochastic techniques. A comparative analysis of the tagging methods is performed using two tagsets (30-tag and 11-tag) with accuracy measures. Brill combined with CRF shows the highest accuracy of 91.83% (for 11 tagset) and 84.5% (for 30 tagset) among all the tagging techniques.

Keywords: Natural language processing, Part-of-speech tagging, POS tagset, Training, Evaluation.

1 Introduction

POS tagging has significant importance in many NLP applications such as parsing, information retrieval, speech analysis, and corpus development. Moreover, it is used as a pivotal component to build a knowledge base for natural language analyzer. It makes the syntactic parser effective as it resolves the problem of input sentence ambiguity. Tagging of words is significantly useful since they are used as the input in various applications where it provides the linguistic signal on how a word is being used within the scope of a phrase, sentence, or document. POS tagging directly affects the performance of any subsequent text processing steps as it makes the processing easier when the grammatical information about the word is known. Usually, supervised and unsupervised approaches are employed in POS tagging, which are further divided into rule-based, stochastic based, and transformation based methods. The rule-based POS tagging uses a dictionary or lexicon for taking the possible tags of a word. The stochastic method considers the highest frequency or probability value to assign a POS tag. Few stochastic tagging methods such as N-grams, CRFs, and HMMs have been implemented for Bengali, English and other languages [1], [2], [3]. The transformation-based method combines rule-based and stochastic techniques such as Brill Tagger.

Designing a POS tagger is a very challenging task for a resource poor language like Bengali. POS tagging of the Bengali sentence is complicated due to its complex morphological structure, the dependency of the subject on verb infections, person-verb-tense-aspect agreement and the scarcity of pre-tagged resources [4], [5]. Moreover, the ambiguity of a word with multiple POS tags and the lack of availability of language experts in Bengali language posses other obstacles that need to overcome. Most of the previous works on POS tagging in Bengali neither highlighted the tagging effectiveness nor investigated their appropriateness. Thus, to address this issue, this paper empirically investigates the performance of 16 POS tagging methods using a supervised approach on a corpus containing 7390 Bengali sentences. Comparative analysis in terms of execution time and accuracy are reported, which helps to decide the suitable use of POS tagging technique for various language processing tasks in Bengali.

2 Related Work

Different approaches have been explored on POS tagging in Bengali and other languages. Stochastic and transformation based methods are the most widely used techniques where a large dataset is prerequisite to achieve good performance. Hasan et al. [6] showed a comparative analysis of n-grams, HMM and Brill transformation-based POS tagging for south Asian languages. A tagset of 26 tags used for Bengali, Hindi and Telegu languages which gained 70% accuracy for Bengali using Brill tagger. Another work implemented the trigram and HMM tagging methods [7] for the Marathi language. A comparison between the stochastic (HMM, Unigram) and transformation based (Brill) methods is presented by Hasan et al. [8]. This work used a small training set of 4048 tokens in Bengali and experimented with two different tagsets (12-tag and 41-tag). The results revealed that Brill tagger performed better than the other stochastic methods. A stochastic based approach proposed by Ekbal et al. [9] concluded that a maximum entropy-based method outperforms the HMM-based POS tagging method for Bengali. Ekbal et al. [10] developed a POS tagger for Bengali sentences using CRF in the name entity recognition task. PVS et al. [11] showed that CRF, along with transformation-based learning, achieved 76.08% accuracy for Bengali POS tagging.

The supervised tagging methods demanded a large amount of tagged data to achieve high accuracy. Dandapat et al. [12] used a semi-supervised method of POS tagging with HMM and Maximum Entropy (ME). Hossain et al. [13] developed a method that checked whether the construction of the Bengali sentence is valid syntactically, semantically or pragmatically. They designed a rule-based algorithm using context-free grammars to identify all POS meticulously. Roy et al. [14] developed a POS tagger that identifies 8 POS tags in Bengali using grammar and suffix based rules. However, they only considered word-level tag accuracy, which failed to identify the correct tag sequence in sentences. Sakiba et al. [15] discussed a POS tagging tool where they used a predefined list of POS tags and applied rules to detect POS tags from Bengali texts. Their method used a very small data set containing 2000 sentences, and it faced difficulties due to the limited rules. Chakrabarti et al. [16] proposed a POS tagging method using the layered approach. Rathod et al. [17] surveyed different POS tagging techniques such as rule-based, stochastic, and hybrid for Indian regional languages where the Hybrid method performed better. Most of the previous approaches of POS tagging in Bengali experimented on a smaller dataset which limits to investigate their effectiveness on evaluation concerning diverse tagsets. In this work, a bit larger dataset consisting of 7390 sentences are used.

3 Methodology

The key objective of our work is to investigate the performance of different types of POS tagging techniques under supervised approach. To serve our purpose, we used a tagged corpus in Bengali developed by Linguistic Data Consortium [18]. The overall process of the work consists of five significant steps: tokenization, training/testing corpus creation, POS tagger model generation, tagged sentence generation, and evaluation. Figure 1 illustrates the abstract representation of the overall process to evaluate POS taggers.

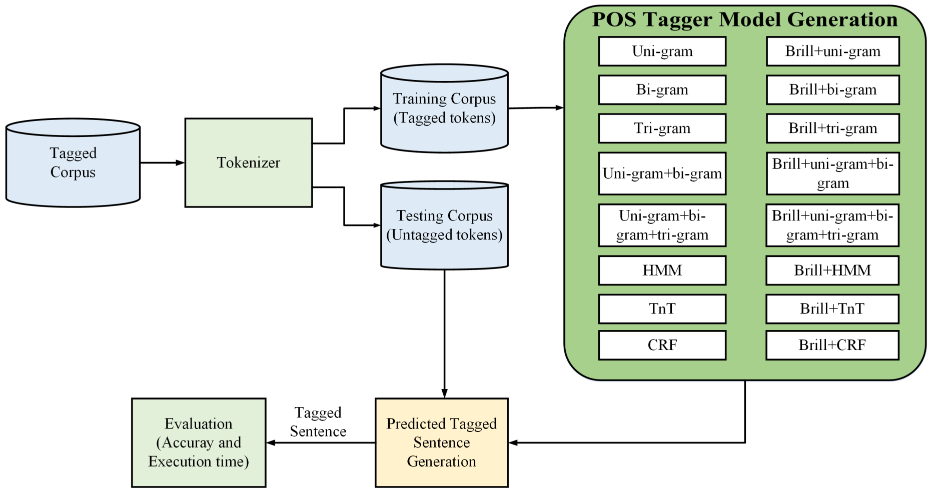
POS Tagger Model Generation
Uni-gram
Brill+uni-gram
Bi-gram
Brill+bi-gram
Tri-gram
Brill+tri-gram
Uni-gram+bi-gram
Brill+uni-gram+bi-gram
Uni-gram+bi-gram+tri-gram
Brill+uni-gram+bi-gram+tri-gram
HMM
Brill+HMM
TnT
Brill+TnT
CRF
Brill+CRF
Tagged Corpus
Tokenizer
Training Corpus (Tagged tokens)
Testing Corpus (Untagged tokens)
Evaluation (Accuray and Execution time)
Tagged Sentence
Predicted Tagged Sentence Generation

Figure 1. Abstract view of POS tagging evaluation process.

3.1 Tagged Corpus

The corpus consists of 7390 tagged sentences with 22330 unique tokens and 115K tokens overall. Two different levels of tagsets have been used for the annotation: 30-tags and 11-tags [19]. The corpus originally tagged with a 30-tag which denotes the lexical categories (i.e., POS) and sub-categories. This 30-tag is mapped into an 11-tag using a mapping dictionary, which considered the lexical categories alone. An extra tag ('Unk') is used for handling unknown words. If a tagger identifies a word which is not available on the training set, then the tagger labels it into "Unk" tag. The table 1 illustrates the tagsets with their tag name and number of tags.

Table 1. Summary of POS tagsets

11 Tagset	**Tagset Count**	**30 Tagset**	**Tagset Count**
Noun (N)	44425	Common Noun (NC)	30819
		Proper Noun (NP)	7994
		Verbal Noun (NV)	2985
		Spatio-Temporal Noun (NST)	2627
Verb (V)	14292	Main Verb (VM)	12062
		Auxiliary Verb (VAUX)	2230
Pronoun (P)	6409	Pronominals (PPR)	5137
		Reflexive (PRF)	362
		Reciprocal (PRC)	15
		Relative (PRL)	448
		WH Pronoun (PWH)	447
Nominal Modifier (J)	14332	Adjective (JJ)	9377
		Quantifier (JQ)	4955
Demonstrative (D)	2876	Absolute Demostrative (DAB)	2421
		Relative Demostrative (DRL)	400
		WH Demostrative (DWH)	55
Adverb (A)	3965	Adverb of Manner (AMN)	1995
		Adverb of Location (ALC)	1970
Participle (L)	573	Verbal Participle (LV)	72
		Conditional Participle (LC)	501
Post Position (PP)	3989	Post Position (PP)	3989
Particle (C)	6704	Coordinating Particle (CCD)	2899
		Subordinating Particle (CSB)	2051
		Classifier Particle (CCL)	324
		Interjection (CIN)	59
		Others (CX)	1371
Punctuation (PU)	13519	Punctuation (PU)	13519
Residual (R)	4348	Foreign Word (RDF)	1873
		Symbol (RDS)	1968
		Others (RDX)	507

3.2 Tokenization

Tokenization is the task of slicing a sequence of character into pieces, called tokens [20]. A token is a string of contiguous characters grouped as a semantic unit and delimited by space and punctuation marks. These tokens are often loosely referred to as terms or words. In our corpus, 7390 tagged sentences were tokenized into a total of 115K tokens with 22,330 unique tokens. A sample tagged sentence and its corresponding tokens are shown in the following.

Sample Sentence: রাজা মহানন্দ রাজধানীতে তৈরি করেছিল শিব মন্দির ও বৈষ্ণবদের মন্দির। (English translation: King Mahananda built the Shiva and Vaishnava temples in the capital.)
Tagged Sentence: রাজা \NP মহানন্দ \NP রাজধানীতে \NC তৈরি \NC করেছিল \VM শিব \NP মন্দির \NC ও \CCD বৈষ্ণবদের \NC মন্দির \NC । \PU
Tagged Tokens: [('রাজা', 'NP'), ('মহানন্দ', 'NP'), ('রাজধানীতে', 'NC'), ('তৈরি', 'NC'), ('করেছিল', 'VM'), ('শিব', 'NP'), ('মন্দির', 'NC'), ('ও', 'CCD'), ('বৈষ্ণবদের', 'NC'), ('মন্দির', ' NC '), ('।', 'PU')]

3.3 Train/Test Corpus Creation

The tokenized dataset is divided into two sets: train corpus and test corpus. The training corpus consists of 98K tagged tokens, while the test corpus contained 17K tokens. The data in the test corpus is untagged to use in the testing phase for evaluation. A data sample in the training corpus (with 11-tagset and 30-tagset) and the testing corpus is illustrated in the following.

Training data sample (11-tagset): [('রাজা', 'N'), ('মহানন্দ', 'N'), ('রাজধানীতে', 'N'), ('তৈরি', 'N'), ('করেছিল', 'V'), ('শিব', 'N'), ('মন্দির', 'N'), ('ও', 'C'), ('বৈষ্ণবদের', 'N'), ('মন্দির', ' N '), ('।', 'PU')]
Training data sample (30-tagset): [('রাজা', 'NP'), ('মহানন্দ', 'NP'), ('রাজধানীতে', 'NC'), ('তৈরি', 'NC'), ('করেছিল', 'VM'), ('শিব', 'NP'), ('মন্দির', 'NC'), ('ও', 'CCD'), ('বৈষ্ণবদের', 'NC'), ('মন্দির', ' NC '), ('।', 'PU')]
Untagged sentence: নিজের মনের কথা বলতে পারা একটি বড় গুণ । (English Translation: Being able to say what's on your mind is a great quality).
Testing data sample (untagged tokens): ['নিজের' , 'মনের' , 'কথা' , 'বলতে' , 'পারা' , 'একটি' , 'বড়' , 'গুণ' , '।']

3.4 POS Tagging Model Generation

The training set is used to train different POS tagging methods. Each tagging method is trained on training corpus tagged with 11 and 30 tagsets—each of 16 POS tagging methods used in a unique way to generate their corresponding training models. N-gram, HMM, TnT, and CRF tagger models generate feature matrices which is used in calculating the probability of the tags. The Brill tagger generates rules used to estimate tags, and the HMM model makes a transition probability matrix called Markov Matrix. An N-gram tagger follows the Bag of Words approach while the CRF tagger uses a statistical approach.

N-gram Tagger

N-gram is a sequence of N words in a sentence. N-gram tagger considers the tags of previous $(n-1)$ words (such as one word in bigram and two words in trigram) in the sentence to predict the POS tag for a given word [7]. The best tag is ensured by the probability that the tag occurred with the $(n-1)$ previous tags. Thus, if τ_1, τ_2 ...τ_n are tag sequence and ω_1, ω_2, ..., ω_n are corresponding word sequence then the probability can be computed using the Eq. 1.

$$P(\tau_i|\omega_i) = P(\omega_i|\tau_i).P(\tau_i|\tau_{i-(n-1)}, ..., \tau_{i-1}) \tag{1}$$

Where, P($\omega_i|\tau_i$) denotes the probability of the word ω_i given the current tag τ_i, and P($\tau_i|\tau_{i-(n-1)}$,...,τ_{i-1}) represents the probability of the current tag τ_i given the (n-1) previous tags. This provides the transition between the tags and helps to capture the context of the sentence. Probability of a tag τ_i given previous $(n-1)$ tags $\tau_{i-(n-1)}$,...,τ_{i-1} can be determined by using the Eq. 2.

$$P(\tau_i|\tau_{i-(n-1)}, ..., \tau_{i-1}) = \frac{C(\tau_{i-(n-1)}, ..., \tau_i)}{C(\tau_{i-(n-1)}, ..., \tau_{i-1})} \tag{2}$$

Each tag transition probability is computed by calculating the count occurrences of n tags divided by the count occurrences of the previous $(n-1)$ tags. Different N-gram models can be combined together to work as a combined tagger.

HMM Tagger

In HMM, the hidden states are the POS tags $(\tau_1, \tau_2, ..., \tau_n)$ and the observations are the words themselves $(\omega_1, \omega_2, ..., \omega_n)$. Both transition and emission probabilities are calculated for determining the most appropriate tag for a given word in a sequence.

The overall probability of a tag τ_i given a word ω_i is,

$$P(\tau_i|\omega_i) = P(\omega_i|\tau_i).P(\tau_i|\tau_{i-1}).P(\tau_{i+1}|\tau_i) \tag{3}$$

Here, P($\omega_i|\tau_i$) is the probability of the word ω_i given the current tag τ_i, P($\tau_i|\tau_{i-1}$) is the probability of the current tag τ_i given the previous tag τ_{i-1} and P($\tau_{i+1}|\tau_i$) is the probability of the future tag τ_{i+1} given the current tag τ_i.

TnT Tagger

In TnT, the most appropriate sequence is selected based on the probabilities of each possible tag. For a given sequence of words of length n, a sequence of tags is calculated by the Eq. 4.

$$\arg\max_{\tau_1 \ldots \tau_n} \prod_{i=1}^{n} [P(\tau_i|\tau_{i-1}, \tau_{i-2}).P(\omega_i|\tau_i)]P(\tau_n|\tau_{n-1}) \tag{4}$$

Where, P($\tau_i|\tau_{i-1}$, τ_{i-1}) denotes the probability of the current tag τ_i given the two previous tags τ_{i-1}, and τ_{i-2}. P($\omega_i|\tau_i$) indicates the probability of the word ω_i given the current tag ω_i, and P($\tau_n|\tau_{(n-1)}$) denotes the probability of the tag τ_n given the previous tag $\tau_{(n-1)}$.

CRF Tagger

CRF is a discriminative probabilistic classifier that calculates the conditional probability of the tags given an observable sequence of words. The conditional probability of a sequence of tags T=(τ_1, τ_2,..,τ_n) given a word sequence W=(ω_1, ω_2,.., ω_n) of length n can be calculated by using the Eq. 5.

$$P(T|W) = \frac{1}{Z(w)} \prod_{i=1}^{n} exp\{\sum_{k} \theta_k f_k(\tau_i, \tau_{i-1}, \omega_i)\} \tag{5}$$

Here, f$_k$(τ_i,τ_{i-1},ω_i) represents a feature function whose weight is θ_k, and Z(w) is a normalization factor which is the sum of all possible tag sequences.

Brill Tagger

Brill tagger is a transformation-based tagger, where a tag is assigned to each word using a set of predefined rules. For a sequence of tags τ_1, τ_2,.., τ_n Brill rules can be represented as the Eq. 6. Here, a condition tests the preceding words or their tags and executes the rule if fulfilled.

$$\tau_1 \rightarrow \tau_2 \tag{6}$$

Stochastic taggers can be used as a back-off tagger with the Brill Tagger. In our work, to investigate the effect of back-off tagger on tagging performance, we used uni-gram, bi-gram, tri-gram, uni-gram+bi-gram, uni-gram+bi-gram+tri-gram, HMM, TNT, and CRF tagging methods with Brill tagger.

3.5 Predicted Tagged Sentence Generation

The generated POS tagging model predicts the highest probability of a tag against the token and labels it with the appropriate POS tag. This process reads the untagged tokens and calculates the probabilities of different tags based on the trained tagger model. Stochastic tagger models (such as N-gram, HMM, TnT, and CRF) use the feature matrices to calculate the probability of the tags. The transformation-based (i.e., Brill tagger) model use the generated rules to estimate the probability of the tags. After POS tagging of each token, individual lists of tagged tokens are created for each sentence. Algorithm 1 describes the process of converting the tagged tokens lists into the tagged sentences.

Algorithm 1: Tagged tokens to tagged sentence generation

```
T ← $List\ of\ Tagged\ Tokens$
tagged_sentence ← []                ▷ List initialization
for $t \in T$ do
    $S \leftarrow$ ""                    ▷ Tagged sentence initialization
    for $token \in t$ do
        $S \leftarrow S + token[word] + "\backslash" + token[tag] + "\ "$
    end
    tagged_sentence.append(S);
end
```

Here T denotes a list of tagged tokens of the corresponding sentence, and S represents the tagged sentence list. Every token is a tuple of $'word'$ and corresponding $'tag'$ as $token\{word, tag\}$. The list of tokens is stacked as a sequence of $'word'$ and $'tag'$ to generate a tagged sentence.

As an example, for the untagged testing tokens (illustrated in Sec 3.3), the prediction model generates the tagged tokens and tagged sentence (for 11-tagset), as shown in the following.

Tagged tokens (11 Tagset): [('নিজের', 'P'), ('মনের', 'N'), ('কথা', 'N'), ('বলতে' 'V'), ('পারা', 'N'), ('একটি', 'J'), ('বড়', 'J'), ('গুণ', 'N'), ('।', 'PU')]
Tagged Sentence (11 Tagset): নিজের \P মনের \N কথা \N বলতে \V পারা \N একটি \J বড় \J গুণ \N । \PU

4 Evaluation Results and Analysis

To evaluate the performance of POS tagging technique, we use two parameters: accuracy (A) and execution time (E). The accuracy can be defined as the ratio between the number of correctly tagged tokens and the total number of tokens. The execution time (E) of a tagger can be computed by the addition of time required during training and testing. Experiments were run on a general-purpose computer with an Intel® Core^{TM} i5-5200H processor running at 2.20 GHz, 8 GB of RAM, and Windows 10. NVIDIA GeForce GTX 950M GPU is used with 4GB RAM. Sixteen POS tagging methods are implemented, and their performance is investigated. Two tagsets (11-tagset and 30 tagset) are used with 115K tokens for evaluating the performance of each POS tagging method in terms of accuracy and execution time. Table 2 summarizes the accuracy of the POS tagging techniques.

The analysis revealed that the Brill+CRF model achieved the highest accuracy of 84.5% (for 30 tagset) and 91.83% (for 11 tagset). The tri-gram methods performed poorly in both sets of POS tags. Additionally, it is observed that the accuracy of the taggers increases with the reduced number of tags in the tagset in all cases.

To examine the effect of the various corpus size on the performance of POS taggers, the taggers were trained with different amounts of tokens from the same corpus. The tagged corpus is partitioned into different sizes as train sets such as 10K, 20K, 30K, 40K, 50K, 60K, 70K, 80K, 90K, 100K, 115K. Each model is trained with each partitioned data sets individually and tested over the 17K untagged testing dataset. Figure 2 shows the performance of the different POS tagging methods for various sizes of training corpus using a 30-tagset.

From the Figure, it observed that Brill+CRF tagger has the highest accuracy even when the corpus size is small. Again, both CRF and Brill+CRF tagger reached almost 75% (for 30-tagset) and 85% (for 11-tagset) of accuracy with a 10K tagged set. The accuracy of each method increased sharply with the rise of the data set and becomes almost steady at 100K.

The performance of the tagger also depends on the execution time. The faster the execution time, the better the performance. We have computed the execution

Table 2. Accuracy of 16 POS tagging methods.

POS Tagger	Accuracy(%) for 30 Tagset	Accuracy(%) for 11 Tagset
Uni-gram	71.46	75.88
Bi-gram	7.79	9.67
Tri-gram	5.33	6.21
Uni-gram+bi-gram	72.59	76.35
Uni-gram+bi-gram+tri-gram	72.42	76.12
HMM	75.12	79.22
TnT	72.35	76.39
CRF	84.27	90.84
Brill+uni-gram	72.99	76.49
Brill+bi-gram	60.58	70.37
Brill+tri-gram	59.3	69.57
Brill+uni-gram+bi-gram	72.75	76.55
Brill+uni-gram+bi-gram +tri-gram	72.54	76.23
Brill+HMM	76.04	79.98
Brill+TnT	72.83	76.45
Brill+CRF	84.50	91.83

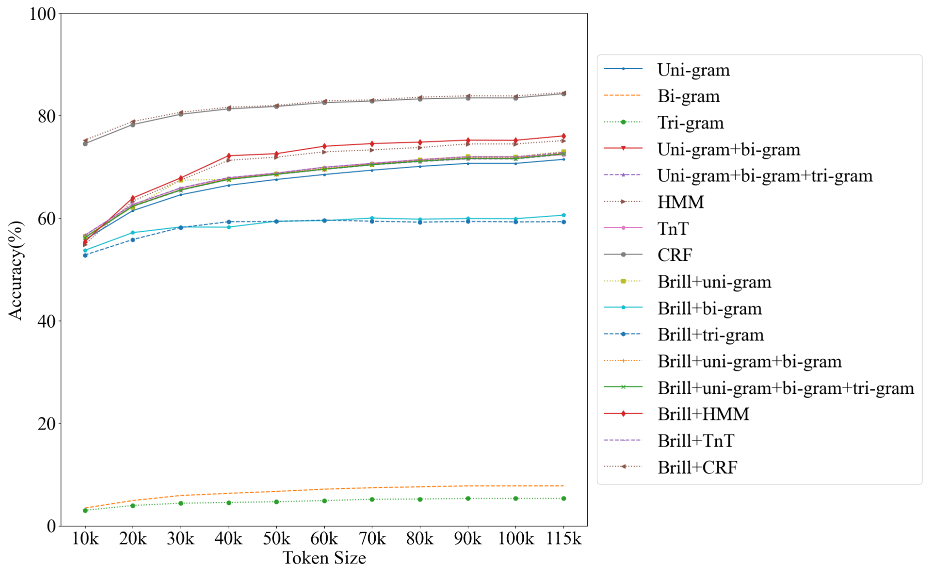

Figure 2. The effect of data size on accuracy for 30-tagset

time of taggers into 11-tagset and 30-tagset. Table 3 shows the performance comparison concerning execution time among 16 tagging techniques.

Table 3. Comparison of execution time among 16 POS tagging methods.

POS Tagger	**30 Tagset**			**11 Tagset**		
	Training Time (s)	**Testing Time (s)**	**Execution Time (s)**	**Training Time (s)**	**Testing Time (s)**	**Execution Time (s)**
Unigram	0.43	0.02	0.45	0.34	0.02	0.37
Bigram	0.6	0.03	0.63	0.59	0.03	0.62
Trigram	0.76	0.04	0.8	0.67	0.03	0.7
Uni-gram+bi-gram	1.0	0.04	1.03	1.04	0.04	1.07
Uni-gram+bi-gram+tri-gram	1.79	0.05	1.84	1.66	0.05	1.71
HMM	0.25	6.88	7.13	0.25	3.8	4.05
TnT	0.53	44.87	45.39	0.5	19.01	19.51
CRF	49.94	0.14	50.08	15.97	0.11	16.09
Brill+uni-gram	21.54	0.16	21.7	16.3	0.2	16.51
Brill+bi-gram	67.08	0.79	67.87	42.97	0.77	43.74
Brill+tri-gram	74.24	0.94	75.17	55.75	0.85	56.6
Brill+uni-gram+bi-gram	12.36	0.2	12.56	12.79	0.21	13.0
Brill+uni-gram+bi-gram+tri-gram	9.64	71.67	81.31	10.62	75.45	86.07
Brill+HMM	39.84	5.76	45.6	25.24	3.5	28.73
Brill+TnT	333.38	44.81	378.19	164.56	22.56	187.12
Brill+CRF	88.56	0.48	89.04	36.83	0.5	37.33

The amount of time required to train a model using the train set decides the training time of the tagger. From Table 3, it is observed that the HMM tagger requires the least training time (0.25s) whereas, Brill+TnT requires the highest training time (333.38s) for 30-tagset. For the 11-tagset, HMM consumed 0.25s and Brill+TnT required 164.56s of training time. In the case, if testing time, uni-gram tagger utilized the lowest tagging time (0.2s) in both tagsets, whereas Brill+unigram+bigram+trigram required the highest tagging time about 71.67s (for 30-tagset) and 75.45s (for 11-tagset) respectively.

The execution time determines the tagging speed of the POS tagging techniques. Fig. 3 shows the execution time required for each tagging methods on our dataset. Results indicate that the Brill+TnT demand more execution time compared to other POS tagging methods.

From the results, it is investigated that Brill taggers, along with other back-off taggers achieved the higher accuracy but they lag in execution time. The Brill+CRF obtained the highest accuracy of 91.83% (for 11 tagset), but it requires a higher execution time (37.33s). On the other hand, the CRF method achieved 90.84% of accuracy and consumes 16.09s for execution. Thus, there is a

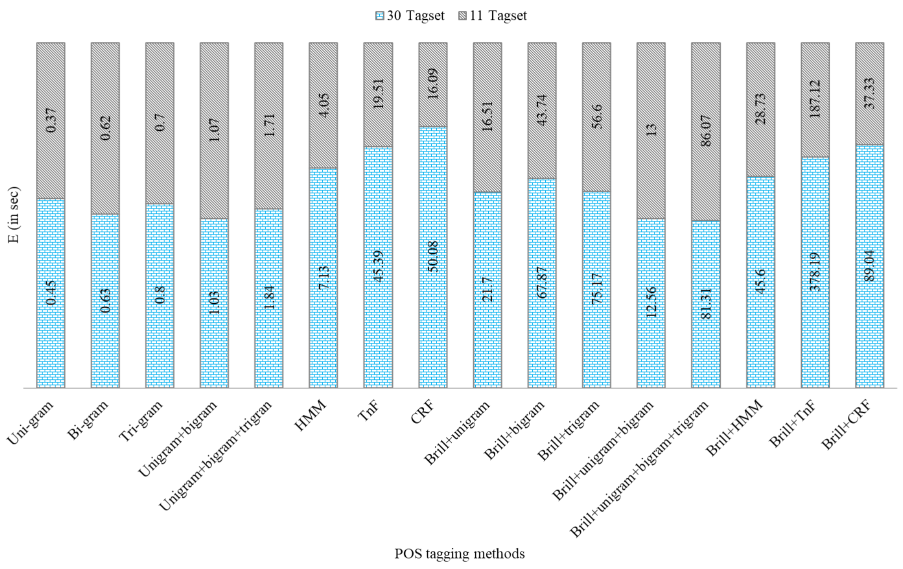

Figure 3. Execution time of different POS Taggers

trade-off between accuracy and execution time. Taking into consideration both the accuracy and execution time, it revealed that the CRF method provided better POS tagging performance compared to other techniques.

5 Conclusion

In this work, we have illustrated and investigated the different POS tagging techniques for the Bengali language. A comparative analysis of 16 POS tagging techniques (eight stochastic-based and eight transformations-based) on a tagged corpus consisting of 1,15,000 tokens have been reported. The comparative analysis revealed that Brill with CRF technique achieved the highest accuracy among other POS tagging techniques, but it requires more execution time. CRF can be maintained as a good trade-off between accuracy and execution time, and this method can be used as a better POS tagging technique. Tagging methods that include both statistical and linguistic knowledge may produce a better performance. The performance of the other tagging techniques such as TAGGIT, CLAWS, Xerox, and Hybrid can be investigated further on the larger tagged corpus with more unique words. These issues will be addressed in the future.

References

1. Dandapat, S., Sarkar, S.: Part of speech tagging for Bengali with Hidden Markov Model. In: Proc. of NLPAI Machine Learning Competition (2006)

2. Diesner, J.: Part of speech tagging for English text data. School of Computer Science Carneige Mellon University Pittsburgh, PA 15213
3. Manju, K., Soumya, S., Idicula, S.M.: Development of a pos tagger for Malayalam-an experience. In: Int. Conf. on ARTCom. pp. 709–713. IEEE (2009)
4. Haque, M., Hasan, M.: Preprocessing the Bengali text: An analysis of appropriate verbs (2018)
5. Bhattacharya, S., Choudhury, M., Sarkar, S., Basu, A.: Inflectional morphology synthesis for Bengali noun, pronoun and verb systems. Proc. of NCCPB 8, 34–43 (2005)
6. Hasan, M.F., UzZaman, N., Khan, M.: Comparison of Unigram, Bigram, HMM and Brill's POS tagging approaches for some South Asian languages (2007)
7. Kumawat, D., Jain, V.: Pos tagging approaches: A comparison. Int. J. of Com. App. 118(6) (2015)
8. Hasan, F.M., UzZaman, N., Khan, M.: Comparison of different pos tagging techniques (N-Gram, HMM and Brill's tagger) for Bangla. In: Adv. and Inno. in Sys., Computing Sciences and Software Eng., pp. 121–126. Springer (2007)
9. Ekbal, A., Haque, R., Bandyopadhyay, S.: Maximum entropy based Bengali part of speech tagging. J. Research in Com. Sci. 33, 67–78 (2008)
10. Ekbal, A., Haque, R., Bandyopadhyay, S.: Bengali part of speech tagging using conditional random field. In: Int. Conf. on SNLP2007. pp. 131–136 (2007)
11. PVS, A., Karthik, G.: Part-of-speech tagging and chunking using conditional random fields and transformation based learning. Shallow Parsing for South Asian Languages 21, 21–24 (2007)
12. Dandapat, S., Sarkar, S., Basu, A.: Automatic part-of-speech tagging for Bengali: An approach for morphologically rich languages in a poor resource scenario. In: Proc. of 45th Annual Meeting of ACL Companion. pp. 221–224 (2007)
13. Hossain, N., Huda, M.N.: A comprehensive parts of speech tagger for automatically checked valid Bengali sentences. In: Int. Conf. ICCIT. pp. 1–5. IEEE (2018)
14. Roy, M.K., Paull, P.K., Noori, S.R.H., Mahmud, H.: Suffix Based Automated parts of speech tagging for Bangla Language. In: Int. Conf. on ECCE. pp. 1–5. IEEE (2019)
15. Sakiba, S.N., Shuvo, M.U.: A memory efficient tool for bengali parts of speech tagging. In: Artificial Intelligence Techniques for Adv. Compu. App., pp. 67–78. Springer (2020)
16. Chakrabarti, D., CDAC, P.: Layered parts of speech tagging for Bangla. Language in India: Problems of Parsing in Indian Languages (2011)
17. Rathod, S., Govilkar, S.: Survey of various pos tagging techniques for Indian regional languages. Int J Comput Sci Inf Technol 6(3), 2525–2529 (2015)
18. Bali, Kalika, M.C., Biswas, P.: Indian language part-of-speech tagset: Bengali. Philadelphia: Linguistic Data Consortium (2010)
19. Sankaran, B., Bali, K., Choudhury: A common parts-of-speech tagset framework for Indian Languages (01 2008)
20. Rai, A., Borah, S.: Study of Various Methods for Tokenization. In: App. of IoT, pp. 193–200. Springer