Utilizing Vector Space Models and Hidden Markov Models for Advanced Part-of-Speech Tagging

Bandaru Rushikesh
(12218364), Jyoshika Rayana
(12216304) *Computer Science
EngeneeringLovely
Professional University*
Jalander, India
ramanjaneyuluj14@gmail.com

Abstract— ***In this study, we investigate a number of models for part-of-speech (PoS) tagging, with a focus on both adaptive and nonadaptive strategies. The adaptive approaches use n-grams from a Hidden Markov Model to evaluate bigram and trigram configurations, as well as three decoding strategies: forward, backward, and bidirectional. Using the Brown Corpus for our studies, we discovered that the bidirectional trigram model approaches state-of-the-art accuracy but has slower decoding times. In comparison, the backward trigram model achieves comparable accuracy while dramatically increasing decoding speed. Our findings imply that examining phrases from last to first improves decoding efficiency. While the backward trigram model performs admirably, we ultimately choose the bidirectional trigram model because of its higher precision in practical applications.***

Keywords—Part of Speech, Hidden Markov Model, rule-based tagger, word structure analysis

I.INTRODUCTION

Part of Speech (PoS) Tagging is an important aspect of natural language processing, and many organizations utilize language processing techniques to provide a variety of applications for their users. Real-world applications include intelligent chatbots, virtual assistants like as Alexa and Siri, social networking platforms, major search engines such as Google, Bing, and DuckDuckGo, as well as a variety of smartphone apps. This study presents an automated system that analyzes English text and accurately determines the part of speech for each word using machine learning techniques.

The system is built on a benchmark text corpus that is utilized for both training and evaluation, as described in Section 2. Section 3 includes a description of the data preparation and preprocessing methods required for the learning algorithm. Section 4 investigates the theoretical elements of the learning algorithms, as well as the modifications made for their use in this context. Section 5 describes the assessment methods used on the model, and Section 6 shows the results of our experiments. Finally, Section 7 reviews the research findings and suggests potential directions for further investigation.

II.DATASET

The dataset used in this study is a collection of text documents that were particularly constructed and tagged to assess the efficacy of supervised learning algorithms in the tagging process. We used the Brown Corpus, which was collected by Winthrop Nelson Francis and Henry Kucera from Brown University's linguistics department. This corpus includes almost a million words from 500 documents. The documents are divided into two categories: informative prose, which has nine subcategories and 374 documents, and imaginative prose, which has six subcategories and 126 documents.

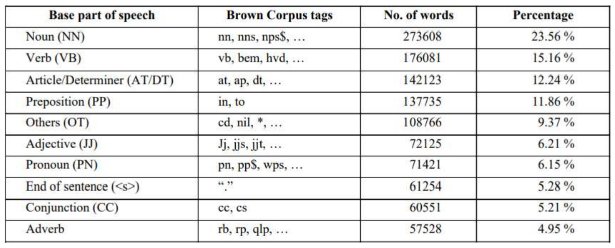
For our analysis, we used two data-splitting strategies: the 70-30 split and k-fold cross validation. In the 70-30 split, documents from each subcategory were randomly assigned, 70% to the training set and 30% to the testing set. The k-fold cross-validation method divides each category into numerous separate groups, or folds. In each iteration, one fold serves as the testing set, while the other folds are used for training, resulting in a more robust evaluation of the model's performance.

III.PREPROCESSING

To properly use the dataset with the learning algorithm, it goes through a number of preparation processes that turn the text into a vector representation, more precisely a bag-of-words model. This format also includes annotated text, which connects each word to its appropriate part of speech (POS). Words and their associated PoS tags are extracted using specified separation criteria; spaces, tabs, and newlines are effective delimiters in the Brown Corpus.

The dataset includes around 100 distinct PoS tags. Analyzing each candidate tag separately would dramatically increase processing time, resulting in slower forecasts and potentially less accurate outcomes. To overcome this issue, we divided the various PoS tags into ten key categories that correspond to the core elements of speech in English language.

Table 1's first column displays the names of the main categories, while the second column contains two or three samples of the original tags found in each category. To extract important words or tokens from the dataset, we first eliminate special characters—such as round brackets '()', square brackets '[]', and braces '{}'—that are not essential for tagging. In addition, we delete tokens that are only numbers. If a token has both numbers and letters, we keep it only if it has a particular amount of remaining letters. During the preprocessing step of the training, we keep two different lists: one for capitalized terms and one for all words reduced to lowercase. The identical filtering method is used on the test set, with the exception that words beginning with a capital letter are left intact. We additionally eliminate duplicate tokens labeled "End of sentence" from the test set because the algorithm does not assess these tags. This step helps to avoid errors during the decoding phase, especially with repeated punctuation marks like "?!?!" that could confuse the processing.



Base part of speech	Brown Corpus tags	No. of words	Percentage
Noun (NN)	nn, nns, nps$, …	273608	23.56 %
Verb (VB)	vb, bem, hvd, …	176081	15.16 %
Article/Determiner (AT/DT)	at, ap, dt, …	142123	12.24 %
Preposition (PP)	in, to	137735	11.86 %
Others (OT)	cd, nil, *, …	108766	9.37 %
Adjective (JJ)	Jj, jjs, jjt, …	72125	6.21 %
Pronoun (PN)	pn, pp$, wps, …	71421	6.15 %
End of sentence (<s>)	"."	61254	5.28 %
Conjunction (CC)	cc, cs	60551	5.21 %
Adverb	rb, rp, qlp, …	57528	4.95 %

Table 1: Base part of speech and the frequency of occurrence for each one of them

IV. Part of Speech - tagging

We used two approaches for the automatic part-of-speech tagging process. The first option is less adaptable, using either the most frequently used word tag as a model, a default tag that assigns the same tag to all words, or a mix of these tactics. The second approach uses Hidden Markov Models (HMMs) to forecast the segment of speech by examining the transition probabilities between hidden states as well as the emission probabilities, which reflect the likelihood of a specific tag being linked with a given phrase.

A. Less adaptive approaches

A default tagger assigns a specific tag to each word in a text corpus, making it a simple approach for tagging. The "noun" tag is typically the most commonly used tag in this paradigm, accounting for approximately a quarter of all tags in the Brown Corpus. Another less adaptable strategy is the most common class baseline model. Using a training set, this approach generates a dictionary of tag frequencies for each word. When assigning a tag to a given word, the model searches the dictionary and chooses the tag with the highest frequency. If the word does not occur in the dictionary (or the training set), the model returns a "not found" tag. To improve this, we can combine this model with the default tagger and label unfamiliar terms as "nouns." This combination can improve accuracy for unknown words while also improving the overall performance of the most common class baseline model. We call these less adaptive techniques (rather than unadaptive) since their parameters are dependent on the training dataset and are not universally applicable across all contexts; only the "default tag" is regarded a generalizable parameter.

B. Adaptive approaches

1) Hidden Markov Model

a) n-Gram model

The Hidden Markov Model (HMM) connects observable states (in this case, the words in the text) with hidden states (part-of-speech tags). An HMM is made up of two important components: matrix A, which represents the transition probabilities, and matrix B, which represents the emission probabilities. The A matrix represents the possibility that one tag will follow another from the previous stage. If the word "The" appears in the previous stage, the current tag is more likely to be a noun, such as "The car…". These probabilities are derived using the test set and the frequency of each tag sequence in the training set.

To create the transition probability matrix for an n-gram model, the Markov model uses a training set to extract occurrence frequencies. Each sequence in the training set contributes to the transition matrix A, either by creating new counts or increasing existing ones. After establishing the Hidden Markov Model during the training phase, we use specific formulas to calculate probabilities for unigrams (1-gram), bigrams (2-gram), and trigrams (3-gram) from the testing set.

Unigram:

$$P(ti) = c(ti) \backslash N \tag{1}$$

where $c(ti)$ represents the occurrence frequency for the tag ti in the training set and N is the total numbers of tokens in the training set.

Bigram:

$$P(ti|ti{-}1) = c(ti{-}1, ti) \backslash c(ti{-}1) \tag{2}$$

where $c(ti{-}1, ti)$ represents the occurrence frequency for the tag $ti{-}1$ followed by the tag ti in the training set and $c(ti{-}1)$ represents the occurrence frequency for the previous tag in the training set.

Trigram:

$$P(ti|ti{-}1, ti{-}2) = c(ti{-}2, ti{-}1, ti)\ c(ti{-}2, ti{-}1) \tag{3}$$

where $c(ti{-}2, ti{-}1, ti)$ represents the occurrence frequency for the tag $ti{-}2$ followed by the tag $ti{-}1$ which is also followed by the tag ti in the training set. c(ti−2,ti−1) represents the previous bigram frequency. The B matrix, with the emission probabilities, represents the probability that a certain tag is associated with a certain word in the training set. The formula, used by the system, to describe the maximum probability estimate is as follows:

$$P(wi|ti) = c(ti, wi)\ c(ti) \tag{4}$$

where $c(ti, wi)$ represents the occurrence frequency for the tag ti associated with the word wi in the training set and $c(ti)$ represents the occurrence frequency for the tag ti in the training set.

b) Smoothing techniques

These strategies are used when specific probability sequences are missing from the A matrix, and the likelihood of missing sequences increases as the rank of the chosen n-gram increases. To overcome this issue, several data smoothing techniques are used in the literature, which allow for the estimation of missing n-gram values using a smoothing function. In this article, we provide two estimating strategies for dealing with missing probability sequences. One of these methods is linear interpolation.

probabilities This method involves creating a new probability by taking a weighted total of the transition probabilities from the unigram, bigram, and trigram models, and multiplying each component by a predefined weight:

$$PLI(t3\ |t1,t2\ ) = \lambda 1P(t3\ ) + \lambda 2P(t3\ |t2\ ) + \lambda 3P(t3|t1,t2) \qquad (5)$$

The values of the coefficients $\lambda 1$, $\lambda 2$, $\lambda 3$ are estimated by the linear interpolation function [2], [5] and the probabilities are computed using formulas (2), (3) and (4). Another estimation method used is the additive smoothing (α-estimate) [6]. This involves adding some constant values to the numerator and denominator in the probability function (usually correlated with the length of the dataset).

$$\theta i = xi + \alpha\ N + \alpha d \qquad (6)$$

Where $xi\ N$ represents the non-smoothed probability, α is the smoothing constant and d is the size of the data (i = 1,). For α = 0, then the above formula (6) does not use any smoothing and for α = 1, then the new formula will be called Laplace's Rule of Succession or Laplace's smoothing technique.

c) Unknown words model

The Hidden Markov The model for unigrams, bigrams, and trigrams can reach reasonable accuracy in the tagging system, but it seldom achieves high accuracy. This limitation is caused by the inclusion of many terms that were not included in the training set, preventing the system from predicting these unknown words. When used to real-world data, in which the quantity of potential words far outnumbers those learned, the system's performance can suffer dramatically.

To overcome this issue, we improved our system's ability to anticipate terms that did not appear in the training set. Tagging unknown words can be done using a variety of ways, including rule-based systems, unsupervised learning algorithms, and word structure analysis algorithms. In this research, we offer a system that uses two ways to tag unknown words. One method focuses on examining the word's structure, while the other uses manually created criteria based on a study of the training data. The final function combines these two methods and returns the chance of connecting the unknown term with a given tag. The structure analysis method seeks to find any relevant suffixes and prefixes that may present in the supplied word. Instead of deriving them from the training set, which might be computationally expensive and produce substandard results, we chose the most representative suffixes and prefixes from a given list [7]. Based on this analysis, the following prefixes and suffixes were selected for the system:

:

List of prefixes: "inter", "intra", "mis", "mid", "mini", "dis", "di", "re", "anti", "in", "en", "em", "auto", "il", "im", "ir", "ig", "non", "ob", "op", "octo", "oc", "pre", "pro", "under", "epi", "off", "on", "circum", "multi", "bio", "bi", "mono", "demo", "de", "super", "supra", "cyber", "fore", "for", "para", "extra", "extro", "ex", "hyper", "hypo", "hy", "sub","com", "counter", "con", "co", "semi", "vice", "poly", "trans", "out", "step", "ben", "with", "an", "el", "ep", "geo", "iso", "meta", "ab", "ad", "ac", "as", "ante", "pan", "ped", "peri", "socio", "sur", "syn", "sy", "tri", "uni", "un", "eu", "ecto", "mal", "macro", "micro", "sus", "ultra", "omni", "prim", "sept", "se", "nano", "tera", "giga", "kilo", "cent", "penta", "tech".

List of suffixes: "able", "ible", "ble", "ade", "cian", "ance", "ite", "genic", "phile", "ian", "ery", "ory", "ary", "ate", "man", "an", "ency", "eon", "ex", "ix","acy", "escent", "tial", "cial", "al", "ee", "en","ence", "ancy", "eer", "ier", "er", "or", "ar", "ium", "ous", "est", "ment", "ese", "ness", "ess", "ship", "ed", "ant", "ow", "land", "ure", "ity", "esis", "osis", "et", "ette", "ful", "ify", "ine", "sion", "fication", "tion", "ion", "ish", "ism", "ist", "ty", "ly", "em", "fic", "olve", "ope","ent", "ise", "ling", "ing", "ive", "ic", "ways", "in", "ology", "hood", "logy", "ice", "oid", "id", "ide", "age", "worthy", "ae", "es".

To effectively use these affixes, the model must first identify the tags associated with words containing these affixes in the training set and assess the likelihood of connection with the encountered tag. If an affix does not appear in the training set, additive smoothing will be used. This smoothing is calculated using the formula below:

$$P_{sp}(x_i|t_i) = \frac{c(t_i,x_i)+\alpha}{\sum_{k=1}^{T_n^{x_i}} k+\alpha d} \qquad (7)$$

With the prefix/suffix xi , $\sum\ k\ Tn\ xi\ k$=1 represents the sum of the tag frequencies associated with that prefix/suffix The second component of the tag identification function for unfamiliar words is a set of manually defined rules. In this article, the following rules apply: "Words that start with a capital letter" are more likely to be nouns; "words that contain an apostrophe and end with the letter's'" are very likely to be nouns; "words that contain a hyphen ('-') or slash ('/')" have a higher probability of being compound words of type OT (others) or JJ (adjective); "words that contain an apostrophe and end with the letter 't'" are very likely to be verbs; and "words that contain an apostrophe and end with the sequences 've' or 'll To combine these two methods, the probability of the unknown word being associated with the current tag is calculated based on the suffixes and prefixes linked to it (denoted as PspP_{sp}Psp), while the rule-based probability is determined according to the established conditions for the weights of the rules (denoted as PrP_rPr). These probabilities are then combined to yield the final probability as follows:

$$P(wk\ |ti\ ) = Psp(xi\ |ti\ ) + Pr\ (wk\ |ti\ ) \qquad (8)$$

Following this addition, the result may exceed the probability interval (0,1]. For this, a threshold function will be executed that will round the value to 1.0. Exceeding the limit only suggests that there is a probability of 100% (maximum confidence) that the current tag being tested is also the correct one, usually this value is obtained for the noun tag which in most cases is also the correct tag.

d) Decoder

In the previous section we presented how models are created for known words (emission and transition probabilities) and for unknown words (rule-based and word structure analysis). Next, we present the decoder part of the system, without it the tagger cannot determine the hidden variables sequence (the tag sequence) associated with the observations sequence (the words of a sentence) [2]. The algorithm used to decode a Hidden Markov Model (HMM) through dynamic programming is the Viterbi algorithm. This algorithm processes the states of the trellis either from left to right or vice versa. The general formula for calculating any node at each step in the trellis is given by:

vt(j)=i=1maxNvt−1(i)PLI(t3∣t1,t2)P(wi∣ti)

where vt(j)v_t(j)vt(j) represents the current Viterbi node being processed for tag jjj, and vt−1v_{t-1}vt−1 denotes the Viterbi node processed at the previous time step.

To improve the performance of the HMM, we created three decoding algorithms, the results of which are reported in this article. These methods are referred to as forward (which moves from the first word of the sentence to the end, followed by a backtrack to determine the final tags), backward (which moves from the end of the sentence to the beginning, then backtracks), and bidirectional (which executes both forward and backward methods, followed by a backtrack determined by the maximum value of the final node).

II. Model evaluation

The dataset used in this paper is a pre-labeled corpus, and the test set is also tagged with the appropriate tags. These tags are not engaged in the prediction process; rather, they are used to assess the performance of the learning algorithms during the model evaluation phase. To evaluate the model, we used two ways.

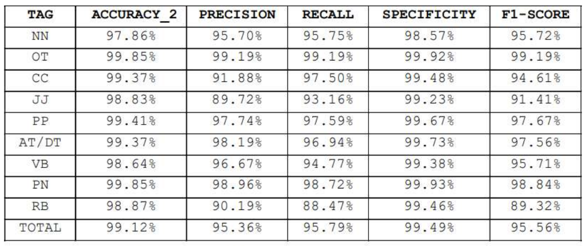
The first method computes a simple prediction accuracy by dividing the percentage of correctly predicted tags (compared to the real ones from the test set) by the total number of predicted tags. We call this evaluation metric "Accuracy_1." In addition, we may calculate the accuracy separately for known and unknown words to acquire a better understanding of each model's performance.

The second approach involves generating a confusion matrix for the entire testing set and extracting evaluation parameters from that matrix, including Accuracy_2, Precision, Recall, Specificity, and F1-score.

Obtained results

Table 2 shows the bidirectional trigram model's results for each component of speech separately, as well as an average for each examined metric (noted in the TOTAL line). This model was chosen because of its superior performance, which combines forward and backward approaches. The results were achieved by splitting the dataset 70-30 and computing the confusion matrix for each tag during the testing phase. The adverb and adjective tags proved the most difficult to forecast, with the lowest scores. These two tags are very context-dependent, complicating their prediction because they may not always coincide with a single tag likelihood in different scenarios. The Others tag, which includes interjections, cardinal numbers, compound words, and similar items with a unique morphological form and are less context-dependent, had the highest prediction score. For example, the word "one" will consistently receive the cardinal number tag.

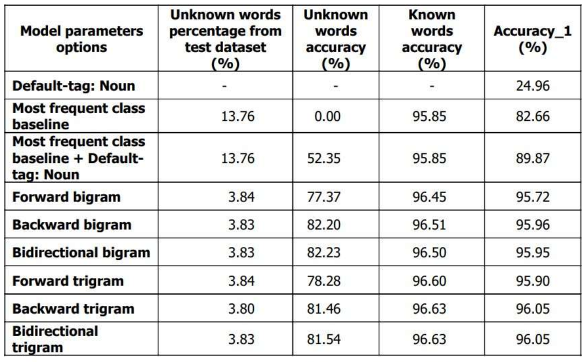
These positive results can be attributed to the fact that the training and test sets are taken from the same dataset and have similar distributions. Even when testing documents differ from those in the training set, they come from the same authors, who tend to employ similar terms in similar circumstances, resulting in consistent part-of-speech designations.



TAG	ACCURACY_2	PRECISION	RECALL	SPECIFICITY	F1-SCORE
NN	97.86%	95.70%	95.75%	98.57%	95.72%
OT	99.85%	99.19%	99.19%	99.92%	99.19%
CC	99.37%	91.88%	97.50%	99.48%	94.61%
JJ	98.83%	89.72%	93.16%	99.23%	91.41%
PP	99.41%	97.74%	97.59%	99.67%	97.67%
AT/DT	99.37%	98.19%	96.94%	99.73%	97.56%
VB	98.64%	96.67%	94.77%	99.38%	95.71%
PN	99.85%	98.96%	98.72%	99.93%	98.84%
RB	98.87%	90.19%	88.47%	99.46%	89.32%
TOTAL	99.12%	95.36%	95.79%	99.49%	95.56%

Table 2: Results obtained by the bidirectional trigram model

For the forward bigram model, the average training time of the model is approximatively 1,41 minutes and the average decoding time for all sequences is 1,44 minutes. For the bidirectional trigram model, the average training time of the model is 1,38 minutes (similar to the average training time for the forward bigram) and the average decoding time for all sequences is 3,56 minutes (twice as long as the forward bigram model). The training time between these models does not differ too much because the training function uses parallel threads to use the most of the processor's capabilities. The decoding time is much longer for the bidirectional trigram model because it has to calculate the trigram transition probability (also the transition probabilities for bigram and unigram) and evaluate the model both forward and backward, after which it needs to decode the best sequence for each sentence. Processing times were estimated on a desktop system with Windows 10, a Quad-Core processor with a frequency of 3.60 GHz and 16 GB Ram.



Model parameters options	Unknown words percentage from test dataset (%)	Unknown words accuracy (%)	Known words accuracy (%)	Accuracy_1 (%)
Default-tag: Noun	-	-	-	24.96
Most frequent class baseline	13.76	0.00	95.85	82.66
Most frequent class baseline + Default-tag: Noun	13.76	52.35	95.85	89.87
Forward bigram	3.84	77.37	96.45	95.72
Backward bigram	3.83	82.20	96.51	95.96
Bidirectional bigram	3.83	82.23	96.50	95.95
Forward trigram	3.84	78.28	96.60	95.90
Backward trigram	3.80	81.46	96.63	96.05
Bidirectional trigram	3.83	81.54	96.63	96.05

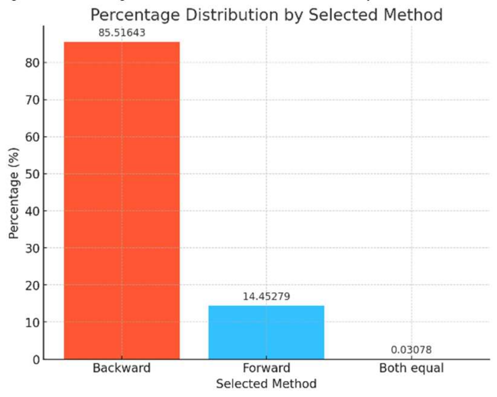
Table 3: Comparatively results obtained by all approaches (adaptive and less-adaptive)

In table 3 we compare the results between the less-adaptive approach and the adaptive approach. The results are presented in terms of simple accuracy (Accuracy_1) for the known and unknown words but also for all the words all together. The known words are words in the test set that also appear in the training set and the unknown words are words that do not appear in the training set. The accuracy on unknown words verifies the tagging system performance that must adapt to unknown situations when it encounters new cases. Those results were obtained using the k-fold cross validation approach to split the dataset (in this case k = 4). For less adaptive models (specifically most frequent class baseline model from table 3), we do not keep a separated list for the words that start with a capital letter. The default tagger has no idea of known or unknown words, and it just forecasts one tag—the most common class baseline. This model, when used with the default tagger, keeps a list of unmodified terms, guaranteeing that capitalized words are not transformed to lowercase in the test set. Without this distinction, and without a separate list of capitalized words, the percentage of unfamiliar words for less adaptive models is substantially higher than that for adaptive models. Less adaptive techniques can produce acceptable results for familiar terms, but their performance may suffer significantly when applied to completely new datasets. Our investigations showed that the backward trigram model is approximately as effective as the bidirectional trigram model. When using the bidirectional technique, the backward branch is selected 85.52% of the time.

Figure 1 shows the percentage of occurrences in which the bidirectional trigram model chooses the backward approach as the optimal backtrace path over the forward method. In only 0.03% of circumstances, both models provide the same outcome due to identical end node values. Look for adaptive models. Interestingly, the bidirectional bigram model has the highest accuracy for unknown words (82.23%), but it does not match the accuracy of the bidirectional trigram for known words (96.63%). The forward bigram model performs badly when compared to other Markov-based models, whereas the forward trigram model outperforms it. The difficulty of part-of-speech tagging is exacerbated by the fact that 67% of tokens in the dataset are ambiguous, able to represent more than one part of speech depending on the context. Using a less adaptive strategy, such as combining the most frequent class baseline with a default noun tag, we may reach an accuracy of roughly 89.87% for correctly predicted tags. According to the preprocessing section, the noun tag accounts for approximately 23.56% of the dataset.

If we remove the end-of-sentence tag, a model relying solely on the default noun tag would yield an accuracy of around 25%, implying that nearly a quarter of the test set was "predicted" correctly. Given the extensive dataset, the differences in results among these models are minimal, which is also reflected in the outcomes presented in Table 2. The issue becomes more

Figure1 : Percentage for each final node value chosen by the bidirectional method



complicated when the testing set have a different distribution than the training set (for example sentences from real world). In table 3, in the first column "Unknown words percentage from test dataset" the percent differs (is between 3.80% to 3.84%) because we use k-fold cross-validation for each test and the documents were randomly grouped in each fold.

III. Conclusions

The purpose of this work is to assess the performance of the Hidden Markov Model (HMM) in a part-of-speech (PoS) tagging system. To do this, we developed and evaluated two distinct HMM models (bigram and trigram) as well as three techniques for deriving maximum probability values for a given sentence (forward, backward, and bidirectional). Our findings show that the trigram model performs better when paired with either the bidirectional or backward methods. Notably, in the bidirectional approach, the backward method was chosen as the best option in almost 85.51% of cases. We also compared the performance of HMM techniques to less adaptive tactics, such as continually forecasting the same tag or relying on the most common class baseline. This comparison sought to show the improvement in accuracy attained by including a learning process into PoS tagging. The accuracy improved from 89.87% (using a combination of less adaptable approaches) to 96.05% (when utilizing the learning strategy). To summarize, the automatic part-of-speech tagging system presented in this research uses existing algorithms in machine learning and natural language processing to attain cutting-edge performance. These findings were achieved with a test set derived from the same dataset as the training set. Furthermore, the system has been tested with user-input texts, with assessments completed entirely by users. Currently, the system only supports English, but it may easily be adapted to handle additional languages.

References

[1] W. Nelson Francis and Henry Kučera at Department of Linguistics, Brown University Standard Corpus of Present-Day American English (Brown Corpus), Brown University Providence, Rhode Island, USA, korpus.uib.no/icame/manuals/BROWN/INDEX.HTM [2] Dan Jurafsky, James H. Martin, Speech and Language Processing, third edition online version, 2019 [3] Lawrence R. Rabiner, A tutorial on HMM and selected applications in Speech Recognition, Proceedings of the IEEE, vol 77, no. 2, 1989 [4] Adam Meyers, Computational Linguistics, New York University, 2012 [5] Thorsten Brants, TnT - A statistical Part-of-speech Tagger (2000), Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000, 2000 [6] C.D. Manning, P. Raghavan and M. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008 [7] Lois L. Earl, Part-of-Speech Implications of Affixes, Mechanical Translation and Computational Linguistics, vol. 9, no. 2, June, 1966 [8] Daniel Morariu, Radu Crețulescu, Text mining - document classification and clustering techniques, Published by Editura Albastra, 2012