Enhancement to N-gram Word Prediction System

Guorui (Gallen) Xu, Echo Hong, Congda Xu and **Jiayi Feng**

Instructor: Adam Meyers
Courant Institute of Mathematical Sciences
New York University
New York, NY 10003

Abstract

We developed four word prediction systems based on some novel enhancement to our baseline algorithm, the pure N-gram model. The enhancement includes POS-tag prediction, weighted POS-Ngram model, and dynamic corpus selection using topic modeling. The accuracy and keystroke saving evaluation of these four systems indicates that the Topic-Based-Weighted-POS-Ngram model has both the highest accuracy and keystroke saving rate. It achieves a 6% higher accuracy and a 3% higher keystroke saving than the baseline system. Finally, the best model has an accuracy of 28.37% and a keystroke saving of 15%.

1 Introduction

Nowadays, efficiency has become one of the features that people mostly concern. As the technology develops, people spend more time on typing with different electronic equipment instead of handwriting on paper. Thus, improving the typing efficiency is an important field that deserves attention. Specifically, word prediction system makes a great contribution to the improvement of typing efficiency on both computers and smartphones.

A word prediction system takes a single word or multiple words as input and outputs a list of predictions for the next word that users may want to type. As shown in Figure 1, Apple has developed the word prediction system for their system input method.

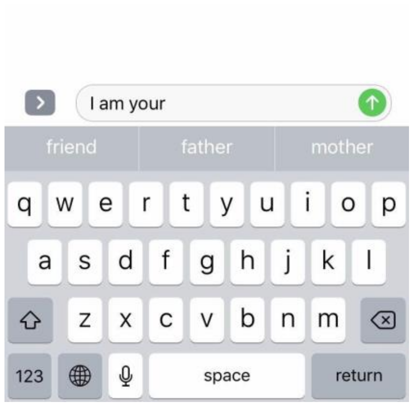
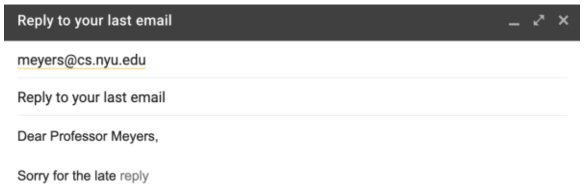


Figure 1: Example of Apple Input System

In addition, Google uses the word prediction for their Gmail input system. In the example of Figure 2, the prediction system takes the word that we have already entered, “Sorry for the late”, and gives the prediction “reply” for the next word that we may want to enter.



Reply to your last email

meyers@cs.nyu.edu

Reply to your last email

Dear Professor Meyers,

Sorry for the late reply

Figure 2: Example of Gmail Input System

Our system will be more similar to the Apple’s word prediction and will present five words for users to choose from. In this paper, we shall discuss the baseline system we choose based on previous works, the algorithm for different enhancements to the baseline system, and evaluations of these enhancements.

2 Background

In order to achieve word prediction, the system generally needs access to some type of language model. A language model captures patterns and uniformity present in natural language and provides an approximative context for upcoming words' prediction. The most traditional way for building word predictor is to use the statistical language modeling, a kind of language model that is based on the probability distribution over sequences of n given words (n-gram) extracted from large corpora (Ghayoomi and Assi, 2005).

For English, Swedish, and many other European languages, there exist considerable word prediction systems today, most based on n-gram. Ghayoomi and Assi's chose, in their research for developing a Persian word prediction system, the N-gram word model as their main source of information. In addition, Steffen Bickel, et al. also adapt the N-gram word model to produce word predictor in their article *Predicting Sentences using N-Gram Language Models*. They compared the N-gram model with the instance-based sentence completion method and the N-gram based completion method gains a better precision-recall profile (Bickel, Haider, and Scheffer, 2005).

However, the basic N-gram word models only take word frequencies into account, especially the unigram statistics, while in most cases, the previous context will significantly influence the probability of the next suggested word. Systems built solely on unigram statistics always propose the same suggestions after a particular word. For higher order n-grams, even though more previous words are taken into account for probability computation and, as a consequence, more syntactic and semantic dependencies are captured, they are still likely to make inappropriate suggestions syntactically and semantically. Hence, we seek to explore some possible adjustment and modification to circumvent the n-gram's weakness. We add part-of-speech tags into consideration because POS tags can group many different word forms into a single word class. In this way, we are able to capture and represent more contextual dependencies. Further, we implement a topic-adapted language model to select appropriate topic corpus dynamically for the given input, whereby the word predictor adapts to the current text.

3 System Development

Previous literatures about commonly used methods for word prediction indicate that most of the systems use N-gram model as their baseline. Due to the straightforward intuition of the model, we also chose N-gram model as our baseline system. Believing that POS-tag of the next word can help the system predict more accurately, we took POS-tag into consideration and used Max Entropy model to predict the POS-tag of the next word. In addition, we speculate that using the corpus of the same topic as the topic of the user's context significantly increases the accuracy of N-gram model. As a result, the systems run a topic modeling after the user has typed enough words to analyze the topic.

We select Penn Treebank as our corpus since it includes most of casual English conversations and also it has POS-tag for us to analyze. We use WSJ corpus in Penn Treebank to be the general corpus and use Brown corpora in Penn Treebank as corpora for different topics. The simple reason of introducing Brown is that WSJ corpus does not differentiate among topics.

3.1 Baseline System: N-gram Model

N-gram model is widely used in the area of word prediction. N-gram model states that the probability is given by the previous words, so the task of predicting the word can be stated as computing a probability (Wasood, Seyyed, 2016):

$$P(w_n|w_1, \dots, w_{n-1})$$

After computing the probability, the task becomes simply sorting the probability and showing the top five possible words measured in probability. The traditional and previous methods only consider high frequency words that match the most recently typed word but ignore the entire previous context. (Swiffin et al,1985). Based on Markov assumption, only the last few words affect the next word. As a result, we decide to implement a tri-gram model as our baseline system, considering the feasibility and computational power of our development. Before the prediction system runs, we make three probability tables: uni-gram probability table, bi-gram probability table, and tri-gram probability table. Uni-gram probability table counts the word that appears after a certain word in the corpus and converts the frequency into probabilities. Bi-gram model and tri-gram model work similarly but count the word that appears after a certain two-word phrase and a three-word phrase. If the user has just typed one word for the sentence, the system runs based on the uni-gram model. If the user has typed two words, the system runs based on the bi-gram model. Finally, if the user has typed more than two words of the sentence, the system runs on the tri-gram model.

However, restricted to the corpus, some three-word phrases have less than 5 outcomes that have occurred in the corpus. As a result, we use the back-offing to give more choices for our users. When there are less than five words in the probability table, the system back-offs to bi-gram model which takes the last two words as the key and generates the remaining possible words. The system back-offs to uni-gram model when there are still not enough outcomes.

3.2 Unweighted POS-Ngram Model

As mentioned in previous sections, using only N-gram model can be improved so we add POS-tag as an additional feature. The idea is that we first predict the next POS-tag and then probability of a word following a certain phrase becomes a conditional probability given by:

$$P(word|tag) = P(Ngram) * P(tag)$$

where $P(word|tag)$ is the probability of the predicted word being the predicted tag; $P(Ngram)$ is the probability that the word appears after the certain n-word phrase; $P(tag)$ is the probability that the word has the predicted POS-tag.

3.2.1 POS-tag Prediction Using Max Entropy

Since there are only 46 POS tags in all the training corpus, the GIS (Generalized Iterative Scaling) model can generate a relatively accurate output. As a consequence, we trained the GIS model to get the POS tag of the word that we intended to predict.

The first and most important step for the training process is choosing the features. We started by just using the previous word and its POS tag, but the accuracy of prediction reported by the model is only around 20%. Then we tried using the previous two words and their POS tags, and we found an obvious improvement of the accuracy. During experiment process, we found that to maximize accuracy, words need to be taken as many as possible. So the final strategy of choosing features for a certain position is to take all the words before that position and their POS tags until the start of the sentence. For instance, if the sentence is:

E.g. I want to eat.

Then there would be four lines generated in the training features file:

1_word=I 1_POS=PRP VBP

1_word=want 1_POS=VBP 2_word=I 2_POS=PRP TO

1_word=to 1_POS=TO 2_word=want 2_POS=VBP 3_word=I 3_POS=PRP VB

1_word=eat 1_POS=VB 2_word=to 2_POS=TO 3_word=want 3_POS=VBP 4_word=I 4_POS=PRP.

After iterating the whole training corpus and put such features file for training, the GIS model reported an accuracy of 72% for POS tag prediction after running 100 iterations.

Due to the feature selection strategy, we passed all the words that already exist in the sentence and their POS tags for the model to predict the next POS tag. For example, if the current sentence is:

E.g. I am your

the system will first do the word tokenization and then assign each word a tag, so the current sentence will be:

[('I', 'PRP'), ('want', 'VBP'), ('to', 'TO'), ('eat', 'VB'), ('.', '.')].

Eventually, these words and POS tags can be put into the model for the prediction and the system will use this tag as a part of the exact word prediction.

3.2.2 Combine POS-tag and N-gram

Following the formula derived in 3.2, we first sorted the N-gram probability table. To improve the efficiency, we decided to pick 200 words with largest probabilities for further computations. We assign 0 to word which doesn't have the predicted POS-tag. The final answer list is the first five with largest conditional probabilities. The process back-offs and repeats in a similar way when the answer list has less than 5 outcomes.

3.3 Weighted POS-Ngram Model

Because of the varying number of words under different POS tag categories, we assume that the POS tag predicted for the next word will influence the possibility of making the correct word prediction. For example, under the category of DT(Determiner), there are limited choices of possible words, and words such as "the", "a", "this", "that" are highly possible correct prediction results. In contrast, for categories like JJ (Adjective), there are at least 500 frequently used adjectives in English, therefore it is highly likely for the system to make an incorrect prediction. With this intuition that the accuracy of the prediction may be influenced by the POS tag of the next word, we thought that assigning weight to different POS tags could be a possible enhancement to the system.

3.3.1 F-measure

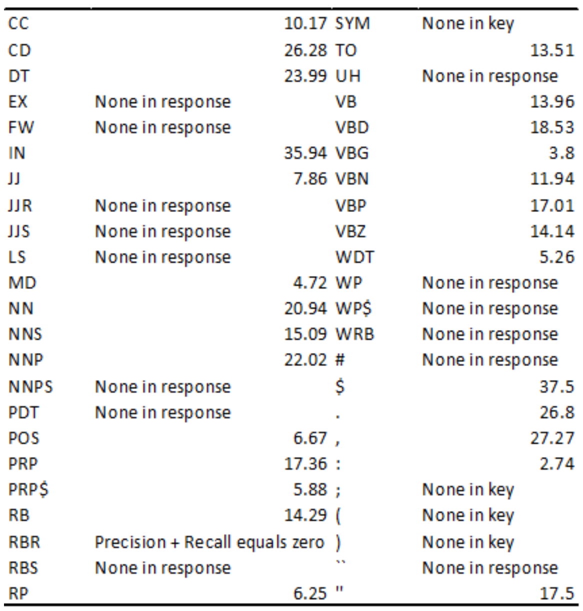
Based on the intuition illustrated above, we calculated the F-measure of prediction result for every single POS tag in Penn Treebank with a development corpus of 1000 words in Wall Street Journal, and the summary is shown in Table 1. From the table, we did see that the F-measure for different POS tags varied largely.

3.3.2 Weighted Score System

To reflect the previous idea, we introduce the weighted score system. The final score is given by:

$$S(word) = P(Ngram) * W(Ngram) + P(tag) * W(tag)$$

For example:



CC	10.17	SYM	None in key
CD	26.28	TO	13.51
DT	23.99	UH	None in response
EX	None in response	VB	13.96
FW	None in response	VBD	18.53
IN	35.94	VBG	3.8
JJ	7.86	VBN	11.94
JJR	None in response	VBP	17.01
JJS	None in response	VBZ	14.14
LS	None in response	WDT	5.26
MD	4.72	WP	None in response
NN	20.94	WP$	None in response
NNS	15.09	WRB	None in response
NNP	22.02	#	None in response
NNPS	None in response	$	37.5
PDT	None in response	.	26.8
POS	6.67	,	27.27
PRP	17.36	:	2.74
PRP$	5.88	;	None in key
RB	14.29	(	None in key
RBR	Precision + Recall equals zero	)	None in key
RBS	None in response	``	None in response
RP	6.25	''	17.5

Table 1: F-measure Summary at development stage

If the user has typed:

I have

The POS-tag predicted is: VBN
And the word predicted by N-gram is:

['a','been', 'taken'],

then the score for a is calculated as

*bi_gram["I have"]["a"] * w_ngram + pos_table["a"]["VBN"] * w_tag*

After deciding the formula of the scoring system, we need to assign weights. Following intuition of different level of importance of different POS-tags, we maintain a POS-tag list based on the result of F-measure taken under development corpus. For those POS-tags with higher accuracy, we assign higher weights to the POS-tags and we assign lower weights to the other POS-tags. Our tests show that **"IN", "CD", "DT", "NNP", "NN"** and punctuations have relatively high accuracy. After testing different combinations of weights, we choose $W(tag)$ =0.4 for tags within the list and $W(tag)$ =0.2 for tags out of the list.

3.4 Topic-Based-Weighted POS-Ngram Model

With the weighted POS-Ngram model, we have taken larger contexts into our system's consideration. As the word choice of a given piece of text may largely depend on its writing topic, we propose that it may be helpful to invoke some short-term learning so as to make the system more appropriate for the specific user. We intend to analyze the current users' temporary typing content, categorize it into the most approximative overall subject and use the selected subject corpus as the upcoming prediction's supplementary guidance.

Further research supports our idea. In Keith Trnka's *Adaptive Language Modeling for Word Prediction*, Keith demonstrates that "topic modeling can significantly increase keystroke saving" for texts from a variety of domains. Topic modeling is a kind of language modeling "that dynamically adapt to testing data, focusing on the most related topics in the training data" (Trnka, 2008), which is exactly what we aimed for.

3.4.1 Topic Grouping with SVM

Because our enhanced system utilizes part-of-speech tags as an additional feature, we need the words in our topic-based corpus to be tagged too. Therefore, we chose the Brown Corpus dataset under the Penn Treebank Tagged Corpora as raw data. The Brown Corpus dataset is divided into 15 Brown subsets that represent different genres and topics, such as popular lore, mystery and detective fiction, letters, love story, etc. We preprocessed the subsets by extracting every tagged word out and formed two different formatted new corpora: one with plain text for classifier's training and one with tags for the prediction system's additional training.

There are numerous methods for automatic document categorization, such as using data mining techniques or machine learning models, among which, the Support Vector Machines (SVM) classifier method is of critical importance. It outperformed four other machine learning text classification methods in a re-examination of text categorization methods conducted by Yang and Liu (1999). Also, Joachims demonstrates SVM's consistent good performance in every experiment in his study (1998). The framework for the SVM classifier is shown in Figure 3.

SVM classifier has the advantage that it is effective in higher dimensional space and still competent in cases where the number of dimensions is greater than the number of samples.

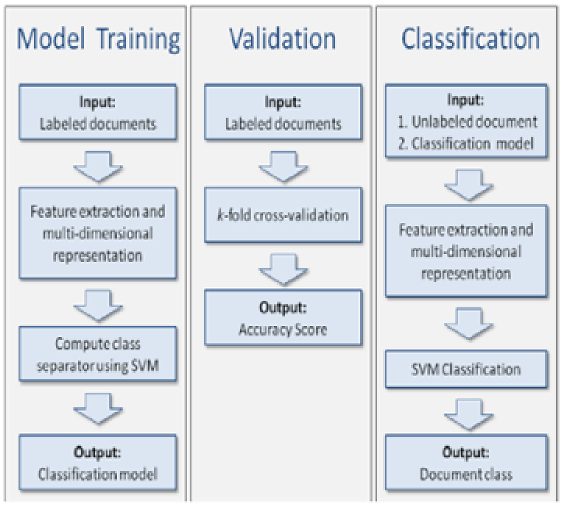

Model Training
Input: Labeled documents
Feature extraction and multi-dimensional representation
Compute class separator using SVM
Output: Classification model
Validation
Input: Labeled documents
k-fold cross-validation
Output: Accuracy Score
Classification
Input: 1. Unlabeled document 2. Classification model
Feature extraction and multi-dimensional representation
SVM Classification
Output: Document class

Figure 3. SVM Classifier Framework (Mertsalov and McCreary, 2009)

These advantages fit exactly what we need because our corpus will very likely to be represented by a matrix with a large dimension of features and a rather small number of samples. We use the Bag-of-Words model to extract features. The model contains all words in the documents. Each unique word in the bag will correspond to a descriptive feature, so presumably, the dimension will be huge. On the other hand, our samples are represented by the number of documents. Since we only have a limited number of documents, the samples will inevitably be smaller. Therefore, the SVM classifier is chosen as our approach to categorize the user's input.

Moreover, when building the Bag-of-Words model, we will eliminate the words in the stop words list in order to abate over-fitting. Also, since Bation et al. suggest that TF-IDF representation yielded the most impressive performance under the SVM classifier, we choose the TF-IDF as the numerical representation for each word features.

The classifier is implemented with Python's scikit-learn library. For any piece of texts given, the classifier will classify it into one of the brown subsets' topics. We separated the plain text Brown corpus into the 80% training set and 20% testing set. After several adjustments, our SVM classifier gains an average accuracy of around 0.75.

3.4.2 Implementation of the Final Model

Due to the concern of efficiency, we preprocessed all of the topic corpus in advance. We made probability tables for the general corpus and other probability table for each topic corpus added to the general corpus. The system will save all the words typed until the user has typed sufficient words for topic grouping (here we decided the number to be 30). The category is determined by the SVM model introduced previously and return the corresponding probability table generated from the corresponding corpus. The rest of the system will predict the next word based on the replaced probability tables.

4 Evaluation

To determine the performance of our word prediction system, we implemented two kinds of evaluation experiments. The first method (Section 4.1) calculates the accuracy of the prediction result. The second one (Section 4.2) calculates the keystroke saving rate, which describes the amount of effort on typing saved by word prediction. Below we denote the four systems we developed System1, 2, 3, and 4 respectively.

4.1 Accuracy

Accuracy indicates what percentage of the words that the user intends to type can be correctly predicted by the system. At the development stage, to test the accuracy of our different prediction systems, we selected 200, 400, 600, 800 and 1000 consecutive lines (each line is a word or space) from Wall Street Journal corpus as our development corpus. We assumed that these words in the development corpus are the content that the user intends to type. Then we ran our system with these development corpus as the answer keys. During the evaluation process, whenever the system made a prediction, the prediction word list at each prediction "slot" would be recorded. After the experiment was completed, we compared the file recording the prediction result with the file recording the answer keys. A prediction will be counted as correct if at each prediction "slot", the prediction word list contains the "intended" word in the answer key. Simply put, accuracy is defined as:

$$Accuracy = \frac{number\ of\ correct\ predictions}{total\ predictions\ made}$$

Comparison of accuracies of the four systems that we developed can be seen in Figure 4, which reveals the relationship between accuracy and corpus size of our four systems. Notice that as the development corpus size increases, the accuracy generally decreases in all four systems.

A direct comparison of the accuracy we achieved for our four systems are presented in table 2. Observe that System 4, which is the Dynamic Topic Choosing and Weighted POS-Ngram model, has the highest accuracy among all the systems.

	Correct	Inorrect	Accuracy
System 1	212	748	22.09
System 2	199	761	20.73
System 3	269	691	28.02
System 4	273	687	28.44

Table 2: Accuracy Summary at development stage

4.2 Keystroke Saving

In addition to accuracy, another evaluation criterion we implemented was keystroke saving (Trnka and McCoy, 2008). Since the goal of a word prediction system is to reduce the number of keystrokes, a primary evaluation for word prediction is naturally how many keystrokes can be saved by the system. The KS equation measures the percentage reduction in keys pressed compared to letter-by-letter text entry.

$$KS = \frac{Keys_{normal} - Keys_{with\ prediction}}{Keys_{normal}} \times 100\%$$

Here we used the same development corpus with size 200, 400, 600, 800 and 1000 as the ones in Section 4.1. We designed a program to achieve the automatic counting of keystrokes with and

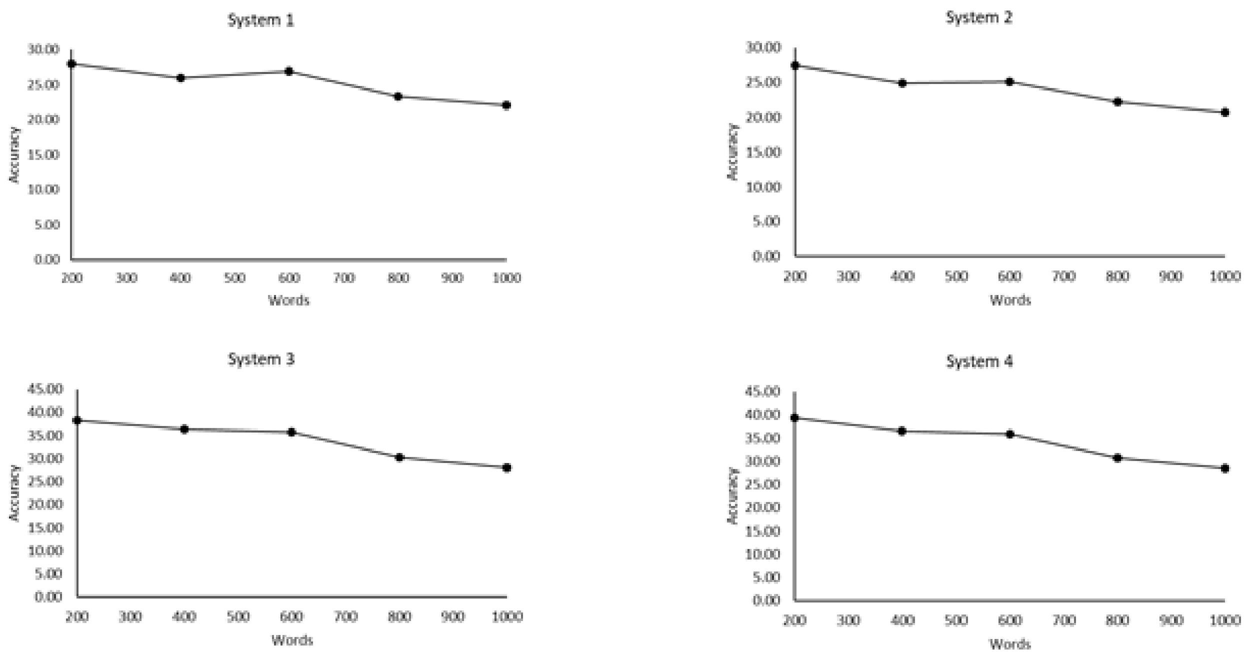


Figure 4: Accuracy Summary for Four System

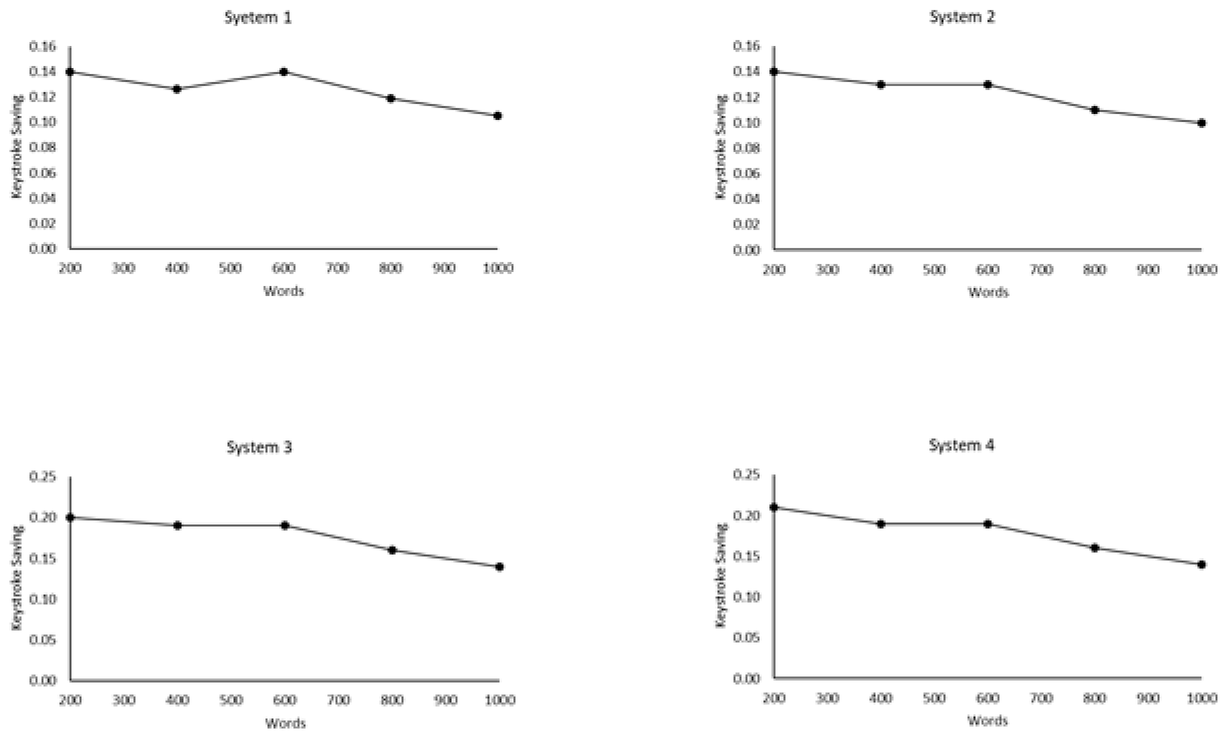


Figure 5: Keystroke Saving Summary for Four Systems

without word prediction. For keystrokes counting without word prediction, we assume that the total number of keystrokes that should be typed by users is the sum of the lengths of all the words in the answer key plus the number of all the spaces between words. For keystrokes with word prediction, we specify that for each correct prediction, the number of keystrokes is one, with the space between the predicted word and previous word automatically provided by the system; for each incorrect prediction, the number of keystrokes is the word length plus one, which indicates that the space between the current word and previous word would be typed. In Figure 5, the summary of keystroke saving of the four systems that we developed is presented. Notice from Figure 5 that as the development corpus size increases, the keystroke saving generally decreases in all four systems.

In Table 3, a direct comparison of the keystroke saving we got for our four systems are presented. We see that our System 4, which is the Dynamic Topic Choosing and Weighted POS-Ngram model has the highest keystroke saving among all the systems.

	Normal Keys	Actual Keys	Keystroke Saving
System 1	5071	4539	0.11
System 2	5071	4566	0.10
System 3	5071	4350	0.14
System 4	5071	4338	0.14

Table 3: Keystroke Saving Summary at development stage

5 Discussion

In the development stage, we have designed four word prediction systems in total. One of them is the N-gram model (System 1) which is our baseline system that is derived from most of the previous works, and the other three systems are Unweighted POS-Ngram model (System 2), Weighted POS-Ngram model (System 3) and Topic-Based-Weighted POS-Ngram Model (System 4). From the evaluation result, we see that in comparison to System 1, System 2 slightly underperfomed in terms of accuracy and keystroke saving. However, System 3's performance is better than that of System 1, with about 6% higher accuracy and 3% higher keystroke saving. As for System 4, the result is slightly better than System 3, achieving the highest accuracy and keystroke saving among all four systems. Based on these results, we conclude that adding a weighted POS tag prediction and topic modeling of text are feasible and effective enhancement to the basic N-gram model in word prediction, and the combination of these three factors can generate a better word prediction performance.

6 Conclusion

In conclusion, we conclude that the Topic-Based-Weighted-POS-Ngram model is the best model we developed so far. Currently, for our test corpus with around 10,000 words, we can achieve an accuracy of around 30% and a keystroke saving of around 15%. In future development, a possible enhancement is to enlarge the training corpus and incorporate corpus with wider topics, to achieve a possibly better word prediction performance.

References

Bation, A. D., Vicente, A. J., & Manguilimotan, E. (2017). Automatic Categorization of Tagalog Documents Using Support Vector Machines. In Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation (pp. 346-353).

Trnka, K. (2008). Adaptive language modeling for word prediction. In Proceedings of the ACL-08: HLT Student Research Workshop (pp. 61-66).

Trnka, K., & McCoy, K. F. (2008, June). Evaluating word prediction: framing keystroke savings. In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers (pp. 261-264). Association for Computational Linguistics.

Bickel, S., Haider, P., & Scheffer, T. (2005). Predicting sentences using n-gram language models. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing.

Ghayoomi, M., & Assi, S. M. (2005). Word prediction in a running text: A statistical language modeling for the Persian language. In Proceedings of the Australasian Language Technology Workshop 2005 (pp. 57-63).

Joachims, T. (1998, April). Text categorization with support vector machines: Learning with many relevant features. In European conference on machine learning (pp. 137-142). Springer, Berlin, Heidelberg.

Mertsalov, K., & McCreary, M. (2009). Document classification with support vector machines. ACM Computing Surveys (CSUR), 1-47.

Yang, Y., & Liu, X. (1999, August). A re-examination of text categorization methods. In Sigir (Vol. 99, No. 8, p. 99).

Maeta, H., & Mori, S. (2012). Statistical Input Method based on a Phrase Class n-gram Model. In Proceedings of the Second Workshop on Advances in Text Input Methods (pp. 1-14).