Next Word Prediction Performance Analysis under Email and Conversational Contexts

Sylvia Chen, Ning Yang, Yuhao Jiang, Shanshan Gong
{sc8043, ny675, yj1412, sg5507}@nyu.edu
New York University

April 2022

Abstract

Next Word Prediction, also known as Language Modeling, is a statistical language modeling method that learns from fragments of user input and output a list of suggested words for users' reference. The traditional way of word prediction focuses on statistical language modeling that measures the probability of a sequence of n given words (n-gram). In this research, we designed a word predictor based on n-gram modeling and implemented it for two common English language contexts: emails and online conversations with different thresholds. Two performance matrices are used to evaluate our system performance, including recall, precision, f-score, hit rate (to measure the average length of suggestion list that contains the correct word). We discuss how a model performs under different natural language contexts and the relationship among the number of n in n-gram, f-score, and thresholds (0, 0.25, 0.5, 0.75).

1 Introduction and Project Motivation

In modern life, email and chat communications tools are part of daily necessity. They are performing similar tasks yet in a different context. Writing emails contains more redundant jobs as many terms and expressions are highly repetitive; chatting on the other hand is more spontaneous and thus less predictable. General word prediction without a specific context is complicated due to the morphological richness of languages; emails, on the other hand, are composed of a certain format and often include specific words or phrases (Mahar and Memon, 2011). Previous works have shed light on next word prediction with the n-gram model under the structure of the Markov models of language, which takes the statistical likelihood of a word given previous n-1 words in order to determine the next word of the sentence. In this paper, we design and analyze n-gram models with a different number of n and implemented them with both email and chat data to see if n-gram models are able to learn from more structured language patterns. We examine to what extent and how n-gram performs differently (from recall, precision, f-score, hit rate) by setting four thresholds (0, 0.25, 0.5, 0.75) for the minimum probability a word needs to meet in order to be placed into the suggestion output for different models. We are using Enron Email Data for the email scenario and Amazon Topical Chat Data for the conversational scenario, with an 8/1/1 split ratio for training, developing, and testing.

2 Related Work

The models we present in the next section draw inspiration from prior work on both probabilistic topic modeling and vector-spaced models for word meanings.

Most of the work related to n-gram language models either use a particular n-gram model for a certain application, or compare the performance and accuracy of different prediction models. Language models alone are powerful tools in provide suggestions and predictions. The bi-gram language model has been widely used in many applications to bring insights to the essential features in lines of words. Recently, Oh and Yi (2021) implemented a bi-gram model to analyze Amazon earbuds' critical features and related sentiment words by evaluating online consumer reviews to help marketers improve products.

Another application of n-gram model is to predict and fit proteins into one of its folds; interestingly, the authors acknowledged a slight increase in the "recognition accuracy" of the bi-gram model in comparison to the uni-gram model (Sharma et al., 2012). Another effort in using multiple n-gram models is encouraged by the analysis of Sindhi words by Mahar and Memon (2011). A closer study of their paper allowed us to come to the realization that a 4-gram model is more efficient for the Sindhi language since it obtains a better perplexity and entropy. Provided with a large enough data set, Sharma et al. were able to achieve a large statistical improvement from their bi-gram and tri-gram model. Inspired by the rise in accuracy, our team considers the approach of using analyzing different n-gram language models.

Moreover, some of the recent works focus on improving the speed, efficiency, and size of the n-gram models. Pauls and Klein (2011) proposed two ways to achieve a faster implementation of the n-gram language model. Both direct-addressing cache and changing the models' interface from word tuples with probabilities to word-and-context encoding with suffix of original query are able to speed up the model however, the model size were not as efficient or ideal. Heafield (2011) also introduces some faster or more space efficient methods to improve the n-gram language model. Mani et al. (2019) has focused on the n-gram application on mobile phone. Due to the necessary speed and storage constraint, Mani et al achieved improvements on the KenLM model (Heafield, 2011) to include a light-weight and improved storage strategy for better speed and space performance. Due to similar motivation and constraint for email composition, we are determined to implement a fast and efficient storage n-gram model.

3 Proposed Approach

3.1 Corpus

In this research, we use Enron emails and Amazon Topical chats for email prediction and conversational prediction respectively. After eliminating trivial or redundant information from both data sets, we tokenized the sentences from emails and messages into words. To achieve a fair comparison, we eliminated the attributes that specific to email data, such as subject, recipient name, and attachment, only using the body part.

As described below, we pre-process the data in a similar manner to Chenet al. (2019), afterwards, we split the data into 10 percent of data used for testing, 10 percent of data used for developing, and the rest 80 percent for training.

1. *Segmentation/Tokenization:* Identifying the beginning and end of each individual sentence, removing all punctuations such as comma, period, examination mark , etc., and finally breaking each sentence into words.

2. *Normalization:* Recognize infrequent words, numbers, etc. in order to exclude or replace them with special tokens.

3. *POS-tagging:* Assign grammatical information of each word in the tokenized document with the outputs in the form of:

```
Traveling\tVBG\n
have\tVB\n
to\tTO\n
a\tDT\n
business\tNN\n
meeting\tNN\n
```

3.2 N-gram models implementation

The n-gram models used to predict the next word are based on the Markov assumption that every node in a Bayesian network is conditionally independent of its non-descendants, given its parents. Therefore, the next word can be predicted using a relatively short history. We count the relative frequencies of words in regard to the previous history of words, calculate the probabilities of each word and follow the estimation of *Maximum Likelihood Estimation* (MLE) to output the most probable one as a prediction To give a reasonable range the history, we limit the number of n to the following four models:

1. n=2 (bigram)

2. n=3 (trigram)

3. n=4 (4-gram)

4. back-off n-gram model

3.2.1 Bigram

Within the bigram model, we calculate the likelihood probability and transition probability depending on one word ahead of it. For likelihood probability, we apply the following equation:

$$P(W_i) = \frac{\text{count(word with the token)}}{\text{count(token)}}$$

For transition probability, we apply the following equation:

$$P(transi) = \frac{\text{count(current token} \mid \text{previous token)}}{\text{count(tokens} \mid \text{previous token)}}$$

After training, we predict the next word depending on the product of the word's likelihood probability and transition probability, given one word.

3.2.2 Trigram

We train our trigram model to fulfill the auto-fill function of our reserach. The algorithm is based on the conditional probability shown below:

$$P(can|trader, a) = \frac{\text{count}(a, trader, can)}{\text{count}(a, trader)}$$

We calculate each word's conditional probability based on previous two words, and store them into a dictionary in the form of:

$$\{'to\,have' : \{'a' : 0.5,' focus' : 0.5\}...\}$$

Using the trained trigram model, we first provide two words as the context for our algorithm. Then, we predict one word at a time, based on the context. For instance, given the context "if you", our model will be able to predict the third word "have", as the conditional probability $P(have|you, if)$ is the highest among all the conditional probabilities of words given "if you".

3.2.3 4-gram

We train a 4-gram model, employing the similar conditional probability as the trigram model:

$$P(have|can, trader, a) = \frac{\text{count}(a, trader, can, have)}{\text{count}(a, trader, can)}$$

And we store each word's conditional probability into a similar dictionary in the form of:

$$\{'a\,trader\,can' : \{'have' : 0.3,' go' : 0.5\}...\}$$

Using trained 4-gram model, we first provide three words as the context, and find the fourth following word after them with the highest conditional probability.

3.2.4 Back-off 4-gram

In this research, we include an non-linear back-off system that is an ensemble of bigram, trigram, and 4-gram models using Katz's back-off model. This system allows model to back off through progressively shorter histories. In our research, the model will start from the 4-gram as the highest order, with the following equation:

$$P(w_i|w_{i-2}w_{i-1}w_{i-3}) =$$

$$\begin{cases} P(w_i|w_{i-1}w_{i-2}w_{i-3}), & \text{if } C(w_iw_{i-1}w_{i-2}w_{i-3}) > K \\ \alpha_1 P(w_i|w_{i-1}w_{i-2}), & \text{elif } C(w_{i-2}w_{i-1}w_i) > K \\ \alpha_2 P(w_i|w_{i-1}), & \text{otherwise} \end{cases}$$

In which, *C(x)* is the number of times x appears in the training, w is the i_{th} word in the given context. K is the minimum number for a word appeared in the n-gram training set in order to consider the conditional probability of a word to be the MLE of that n-gram. α is the back-off weight. In this paper, we adopted pure back-off methods (Robert et al., 2009), where all instances of $\alpha = 1$ for both contexts. As for K, we adopt different thresholds S (0, 0.25, 0.5, 0.75) for comparison, where $S = K/N$, in which N being the length of number of words.

3.3 Evaluation

To evaluate and compare performances between different n-gram models across different textual scenarios, we applied the following attributes (noted that an instance is considered correct if any member of a suggestion list is correct):

- **Precision**: the number of correct results divided by the number of system output. A higher precision indicates better performance.
- **Recall**: the number of correct results divided by the number of test cases. A higher recall implies better performance.
- ***f*-score:**

  $$F = \frac{precision \times recall}{precision + recall}$$

  A higher f-score indicates that the experimental result is more accurate.
- **Hit Rate (HR)**: The percentage of correct words that appear in the suggestion list. A higher hit rate implies better performance.

The f-score is widely used in the field of natural language processing to measure a test's accuracy in the range [0,1]. Also, inspired by the word prediction analysis by Ghayoomi and Assi(2006), in which they introduced Hit Rate(HR), we simplify it accordingly to fit our experiment.

4 Results

The length of the training corpus was 50,000 sentences, which contained 421,952 words and 398,421 words (without space and punctuation) respectively for email and conversational contexts. The length of the testing corpus was 10,000 sentences, which contains 10,000 sentences, which contained 94,382 words and 89,489 words (without space and punctuation) respectively for email and conversational contexts.

The overall performance of both the email and conversational data across different thresholds with various n-gram models reflected a fairly consistent pattern. In the experiment of each model, precision, recall, and f-scores are higher as the level of thresholds decreases. However, the Hit Rate(HR) becomes higher as the level of thresholds increases. Email data as a whole presented a more structured pattern as the number of n is positively related to the f-score performance under email data. Interestingly, bigram and trigram have an overall better performance under conversational data, especially with low thresholds.

	0		0.25		0.5		0.75	
	Email	Chat	Email	Chat	Email	Chat	Email	Chat
F-Score	0.143	0.503	0.083	0.088	0.006	0.005	0.005	
Precision	0.144	0.503	0.160	0.205	0.085	0.194	0.075	
Recall	0.143	0.503	0.056	0.056	0.003	0.002	0.002	
Hit Rate	0.084	0.003	0.528	0.513	1.0	1.0	1.0	

Figure 1: Bigram performance for different thresholds. No system output for chat data above 0.75 threshold.

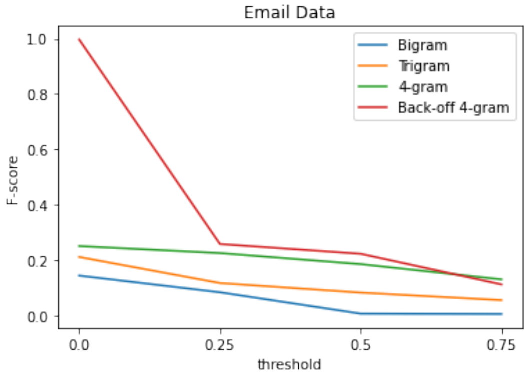
	0		0.25		0.5		0.75	
	Email	Chat	Email	Chat	Email	Chat	Email	Chat
F-Score	0.211	0.372	0.116	0.170	0.082	0.119	0.055	0.068
Precision	0.314	0.461	0.243	0.326	0.261	0.380	0.280	0.380
Recall	0.158	0.312	0.077	0.115	0.049	0.070	0.030	0.037
Hit Rate	0.084	0.188	0.879	0.876	0.984	0.967	0.998	0.999

Figure 2: Trigram performance for different thresholds.

4.1 Email Data

The performance for email data presented a more structured pattern: the increase in the number of n will largely increase the overall precision for all thresholds. The most significant jump is between trigram and 4-gram models (Figures 2 and 3). The precision value increases by more than 217%. Given the increase of n is slightly positively related to recall, the f-score will significantly increase mostly because of the positive correlation between n and precision. Traditionally, recall and precision have offset relationships. However, in our experiment, they have a positive relationship which requires us to do further research about the reason in the future.

The graph below explicitly shows how f-score changes with various thresholds in different models.



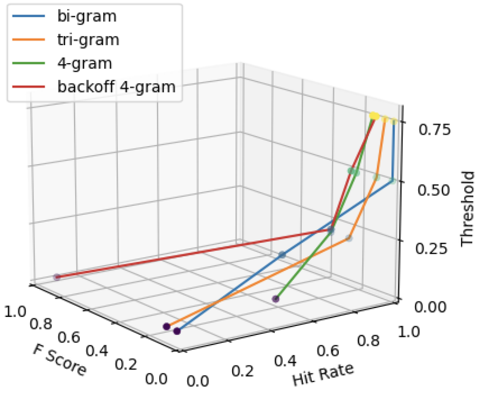
	0		0.25		0.5		0.75	
	Email	Chat	Email	Chat	Email	Chat	Email	Chat
F-Score	0.250	0.089	0.225	0.068	0.185	0.047	0.130	0.031
Precision	0.996	0.997	0.996	0.997	0.996	0.998	0.995	0.998
Recall	0.143	0.047	0.127	0.035	0.102	0.047	0.130	0.031
Hit Rate	0.626	0.513	0.876	0.839	0.967	0.929	1.0	0.999

Figure 3: 4-gram performance for different thresholds.

	0		0.13	0.25		0.5		0.75	
	Email	Chat	Email	Email	Chat	Email	Chat	Email	Chat
F-Score	0.997	0.345	0.262	0.258	0.285	0.222	0.217	0.112	0.145
Precision	0.997	0.998	0.997	0.997	0.997	0.997	0.997	0.995	0.997
Recall	0.997	0.209	0.151	0.148	0.166	0.125	0.122	0.059	0.078
Hit Rate	0.089	0.579	0.867	0.903	0.842	0.971	0.923	1.0	1.0

Figure 4: Back-off 4-gram performance for different thresholds. This system has a dramatic surge regarding to f-score with email data when threshold falls below a certain level; the dividing threshold is around 0.13.

All four models are shown in four colors. As shown in the graph, bigram, trigram, and 4-gram models share similar patterns. That is as n in n-gram models increase, f-score increases smoothly and slowly. However, the back-off 4-gram model has a rapid change in f-score when the threshold falls below 0.13 (Figure 4: the red line's slope becomes steep from 0.25 to 0 thresholds), and f-score reaches a nearly perfect value (1.0) at the end. What's more, from these four lines' positions, we conclude that the back-off 4-gram model has the best performance, the following 4-gram, trigram, and bigram models.

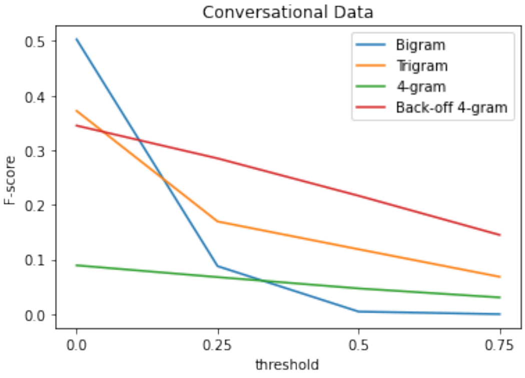


Also, these models share a similar pattern of the hit rate change. The hit rate will increase as the threshold rises. However, only 4-gram has a relatively smooth increase in hit rates. All other three models' hit rates rise more than 52.8% when thresholds increase from 0 (Figures 1, 2, and 4).

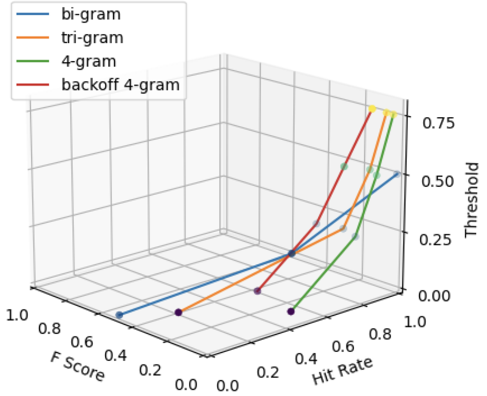
Under the email kind scenario, we assume that users do not wish to have a long suggestion list to choose the word they want when they write emails. In other words, a good prediction test for an email should have a high hit rate and a high precision value. Based on the research results, we will choose a back-off 4-gram model with thresholds higher than 0.13, which has a hit rate of 0.867 and a precision of 0.997. One problem is its low f-score which is a potential research direction for us to improve the performance.

4.2 Conversational Data

The performance for conversational data, comparing with the email one, is statistically less predictable and more disordered. The result of how different model performs with the conversational data are shown in following graph:



Noticeably, bigram and trigram have a better performance in the conversational context whereas 4-gram has a better performance in the email context. One of the potential reasons is that email has more identical sentences such as greetings, salutations, and closing sentences, which works better with 4-gram with relatively high precision. On the contrary, daily conversations contain more colloquial phrases that are short and diverse, which are hard to predict under 4-gram.



The graph above shows the relationship among threshold, f-score, and hit rate. When threshold increases, f-score decreases and hit rate increases; when threshold decreases, f-score increases and hit rate decreases. f-score is a harmonic measurement that balances the trade-off between precision and recall, yet a high f-score usually goes along with a low hit rate. In conversational context, bigram has a high f-score under low thresholds, with a relatively low hit rate as shown in Figure 1. In a conversational context, the hit rate for bigram when the threshold is set to zero is 0.003, which means the system on average needs to output a suggestion list of around 333 words in order to have one word correct. This length of suggestion list apparently is not ideal for daily conversation usage. We use the iOS system and the Android system's next word prediction suggestion length as the ideal prediction length, which are both 3 words for the English context. At threshold 0.25, the hit rate for bigram increased 170% from 0.003 to 0.513, with an 82.5% fall in f-score from 0.503 to 0.088, which means on average the suggestion length decreases to around 2. By inference, we know the ideal threshold lies in the range of 0 to 0.25, with proximity to 0.25. Similarly, we can infer narrow the ideal threshold for other n-gram models: for the lower number of n as bigram and trigram, the ideal threshold lies between 0 and 0.25; for the larger number of n as 4-gram and the back-off 4-gram, the ideal threshold is 0.

5 Conclusions and Future Works

5.1 Real-life Application Based on Observations

In a real-world scenario, email and instant online chatting tools are playing different roles in our life. Email, which is more formal and often used mostly in workplaces, appeared to be more well-structured and has more repetitive patterns. Chat, on the other hand, appeared to be more casual, and spontaneous and contained fewer patterns. As we discussed in the result section, different n-gram models performed differently at different thresholds when dealing with a different type of data context. In the email, given our experience with Google Gmail and Microsoft Outlook, users are only given one prediction at a time. Therefore, we would like to emphasize more the precision with a high hit rate of as close to 1 as possible. As for the chat scenario, users are given a suggested list of words with a length of 3 used by most of the industry standards. We would like to emphasize more the hit rate at around 0.33 with the highest f-score given the hit rate.

5.2 Future Improvements

In the current work, we have focused on experimenting with different n-gram models to predict either the next word in an email - a specific, well-structured kind of message or the next word in a chat message - a less predictive and pattern-lacking kind of message. While we have demonstrated that the capabilities of the four models implemented - bi-gram, tri-gram, 4-gram, and back-off 4-gram - vary in the level of structure provided by the data, we plan to continue improving our models on the following topics:

1. *Context Clue:* Take advantage of the other information provided by each email or message such as the subject and/or the text that he/she is replying to predict a more accurate response (i.e. if the mood of the email or message is negative when the user is typing "I feel", the algorithm will suggest to type "sorry about your"). Inspired by Oh and Yi (2021)'s work on sentiment analysis, we can employ sentiment lexicons: AFINN(Nielsen, 2011) and BING(Liu, 2012).

2. *Personalize:* Acknowledging that each individual has his/her writing style, we can improve the user experience by tuning our n-gram model for each individual to emails or messages they sent in the past few months. Chen, Bansal, et al. (2019) suggest adding a Katz-backoff implementation to the n-gram model, which is easy to train and does not require much data.

3. *Add-One Smoothing:* Since it is very likely to obtain zero for the probability, we hope to include assigning non-zero probabilities to all n-grams. Specifically, we will use the add-one smoothing technique to add one to the count provided that it is easy to implement and successfully avoid the problem of getting zeros.

4. *Profound Evaluations:* Apart from goals listed above, we also plan to apply BLEU as our evaluation metrics to find out what factors affect the correctness of soundness, semantics, and sentiments of the given texts the most.

References

Oh, Yun Kyung, and Jisu Yi, “Asymmetric Effect of Feature Level Sentiment on Product Rating: An Application of Bigram Natural Language Processing (NLP) Analysis.” Internet Research, Emerald Publishing Limited, 30 July 2021, `https://www.emerald.com/insight/content/doi/10.1108/INTR-11-2020-0649/full/html`.

A. Sharma, J. Lyons, et al., “A Feature Extraction Technique Using Bi-Gram Probabilities of Position Specific Scoring Matrix for Protein Fold Recognition.” ScienceDirect, 13 December 2012, `https://doi.org/10.1016/j.jtbi.2012.12.008`.

J. A. Mahar and G. Q. Memon, ”Probabilistic Analysis of Sindhi Word Prediction using N-Grams.” Australian Journal of Basic and Applied Sciences, vol.5, no.5, pp.1137-1143, 2011, `https://www.researchgate.net/profile/Ghulam-Memon/publication/228749971_Probabilistic_Analysis_of_Sindhi_Word_Prediction_using_N-Grams/links/0046353498acb3fc82000000/Probabilistic-Analysis-of-Sindhi-Word-Prediction-using-N-Grams.pdf`.

A. Pauls and D. Klein, ”Faster and smaller n-gram language models” in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, vol.1, pp.258-267, 2011, `https://aclanthology.org/P11-1027/`.

Kenneth Heafield, ”KenLM: Faster and Smaller Language Model Queries” in Proceedings of the Sixth Workshop on Statistical Machine Translation, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics, 2011, `https://aclanthology.org/W11-2123/`.

S. Mani, S. Ghosh, et al., ”Real-Time Optimized N-Gram for Mobile Devices.” 2019 IEEE 13th International Conference on Semantic Computing (ICSC), pp.87-92, 2019, https://arxiv.org/abs/2101.03967.

https://www.overleaf.com/project/625ef46cc2e483081e43c0d2
M. X. Chen, G. Bansal, et al., “Gmail Smart Compose: Real-Time Assisted Writing.” ArXiv.org, 17 May 2019, https://arxiv.org/abs/1906.00080.

M. Ghayoomi and M. Assi, ”Word Prediction in a Running Text: A Statistical Language Modeling for the Persian Language.” Researchgate.net, January 2006, https://www.researchgate.net/publication/228773739_Word_Prediction_in_a_Running_Text_A_Statistical_Language_Modeling_for_the_Persian_Language.

S. Trajanovski, C. Atalla, et al., ”When does text prediction benefit from additional context? An exploration of contextual signals for chat and email messages.” Aclanthology.org, June 2021, https://aclanthology.org/2021.naacl-industry.1.pdf.

A. Mueller, G. Nicola, et al., ”Cross-Linguistic Syntactic Evaluation of Word Prediction Models.” Arxiv.org, 21 May 2020, https://arxiv.org/abs/2005.00187.

Robert C. Moore and Chris Quirk, ”Improved Smoothing for N-gram Language Models Based on Ordinary Counts.” Arxiv.org, January 2009, https://aclanthology.org/P09-2088.pdf.