First-two-char input method

Jules Désir

June 9, 2024

Abstract

In this paper, we implement a text input method, referred to as the First-Two-Char Input Method, designed to enhance typing efficiency and accuracy [1]. This method uses the initial two characters of each word to predict and suggest the complete word, thereby reducing the number of keystrokes required. We used a comprehensive dataset to train and evaluate our model: the Wikipedia raw text English content as of March 1st 2022. The proposed method achieves an accuracy of 0.36 on the test dataset, highlighting its potential for practical applications. This method is a promising approach for text input technology, offering a user-friendly solution for various typing interfaces.

1 Introduction

The advent of mobile technology and compact computing devices has dramatically transformed the way people interact with digital systems. Traditional keyboards, once the primary means of text input, are often impractical for small screens and touch interfaces. Consequently, there is a growing need for innovative input methods that enhance typing efficiency and user experience.

One promising solution is the First-Two-Char Input Method, which aims to streamline the typing process by predicting words based on their initial two characters. This method is inspired by the observation that the first two characters of a word often provide sufficient information to narrow down potential word choices significantly. By leveraging this property, the First-Two-Char Input Method reduces the number of keystrokes required, thus improving typing speed and accuracy.

The motivation behind this research is twofold. Firstly, there is an increasing demand for efficient input methods in mobile and wearable devices where screen space and input precision are limited. Secondly, enhancing typing efficiency can greatly benefit users with physical disabilities or motor impairments, providing them with a more accessible means of communication.

To evaluate the efficacy of the First-Two-Char Input Method, we used a comprehensive dataset encompassing a wide range of linguistic data. This dataset serves as a foundation for training and testing our predictive model. Through experimental evaluation, we demonstrate the method's potential to significantly enhance typing performance.

In this paper, we detail the methodology of our approach, describe the language resources used, and present a thorough analysis of the experimental results. The findings of this paper indicate that the First-Two-Char Input Method offers a practical solution for modern text input challenges.

2 Method

In this section, we describe the methodology used to implement a First-two-char input algorithm. Two different approaches were implemented: a word 1-gram language model and a word 2-gram language model [MMYN99, CL00]. Both methods were designed to predict the most likely word given the first two characters, leveraging different statistical properties of the language.

We describe the data structure used, followed by the two predictive approaches.

[1] All the code is available on Github: https://github.com/julesdesir/First2char-input-method

2.1 Data processing

The initial step involves creating a data structure to facilitate efficient word prediction. This is achieved through 2 functions.

First, the *dictionary*(*words_list*) function maps each pair of initial characters to a list of corresponding words. This allows quick lookup of words based on their initial characters.

Then, the *data_prep*(*dataset*) function processes the input dataset (raw text) by splitting it into individual words and creating the dictionary using the dictionary function.

2.2 First method: word 1-gram language model

The word 1-gram language model relies on the frequency of individual words and the conditional probability of a word given its initial two characters. The following steps outline the approach:

2.2.1 Helper functions

We wrote 2 functions, $f_X(d, x_i)$ and $f_{Y,X}(d, y, x_i)$, which count respectively the occurrences of word x_i in the dictionary d and the occurrences of x_i in the dictionary d given the first two characters noted y.

2.2.2 Probability functions

Two functions are used to compute probabilities. The $P^{Y,X}(d, y, x_i)$ function calculates the conditional probability $P(y \mid x_i)$, which is the likelihood of y given x_i, based on the dictionary d. And the function noted $P^X(words\_list, d, x_i, sumf_X)$ computes the probability $P(x_i)$, which is the likelihood of word x_i occurring in the dataset, with d the used dictionary and $sumf_X$ the sum of the values taken by $f_X(d, x_j)$ for all x_j in *words_list*.

2.2.3 Argmax function

We then implemented the $argmax_{1\_gram} P^{Y,X} P^X(words\_list, d, y)$ function which finds the word x_i that maximizes the product $P(y \mid x_i) \times P(x_i)$ based on the word list *words_list* and the dictionary d.

2.2.4 Final function

Eventually, the $f_{1\_gram}(dataset, concatenated\_2\_char)$ function uses the helper, probability, and argmax functions to predict a sequence of words from a concatenated string of first two characters.

2.3 Second method: word 2-gram language model

The word 2-gram language model incorporates the additional context of word pairs, utilizing bigrams to improve predictive accuracy. We keep the function $P^{Y,X}(d, y, x_i)$ as before but we replace $P^X(words\_list, d, x_i, sumf_X)$ by another function.The approach involves the following steps:

2.3.1 Helper functions

First, the function *count_bigrams*(*words_list*) counts the occurrences of each possible bigram in the given word list.

Then, *calculate_probabilities*(*bigram_counts*): computes the conditional probabilities of each word given the previous word.

And after that, *train_bigram_model*(*words_list*) applies *count_bigrams* and *calculate_probabilities* successively to train the bigram model.

2.3.2 Argmax function

The function $argmax_{2\_gram} P^{Y,X} P^X(words\_list, d, y)$ leverages the previously introduced helper functions to find the word x_i that maximizes the product $P(y \mid x_i) \times P(x_i)$ using bigram probabilities.

2.3.3 Final function

As for the 1-gram method, the function $f_{2\_gram}(dataset, concatenated\_2\_char)$ predicts a sequence of words from a concatenated string of first two characters, but this time by using the trained bigram model. Similarly to the 1-gram model, the 2-gram model is tested using a sample input to demonstrate its predictive capability.

The described methodology provides a comprehensive framework for the first-two-char input algorithm, leveraging both unigram and bigram language models to optimize word prediction accuracy and efficiency.

3 Language resource

This section details the language resources used in the development and evaluation of the First-Two-Char Input Method. The two subsections below describe the dataset and the technical restrictions encountered due to limited computing power.

3.1 Dataset presentation

For the experiments, we used the Wikipedia raw English content as of March 1st, 2022, available through the Hugging Face datasets library [2] [3]. This dataset provides a comprehensive collection of English-language articles from Wikipedia, offering a rich source of text data for training and testing the language models. The sizes of the downloaded dataset files and the generated dataset are 11.69 GB and 20.28 GB, hence a total amount of disk used amounting to 31.96 GB.

Next, we extracted the raw text from the dataset. The text was saved into a `.txt` file to facilitate further processing and model training. The extracted raw texts provided a substantial corpus for developing and testing the input method.

3.2 Technical restrictions due to limited computing power

Despite the rich dataset, we faced technical restrictions due to the limited computing power of my personal computer. Handling the entire Wikipedia dataset in its raw form was infeasible, necessitating the reduction of the dataset size for manageable processing.

To address this, we implemented a function to reduce the raw text file to a specified number of sentences. This allowed us to create a smaller, more manageable dataset suitable for training and evaluation on a less powerful machine.

By reducing the dataset size, we ensured that the experiments could be conducted efficiently within the computational limits of the available hardware. The reduced dataset retained enough linguistic diversity to train and evaluate the models effectively, while being small enough to be processed without overwhelming the system resources.

Yet, higher accuracy levels could be reached with more training data.

The described methods and technical adjustments enabled the successful implementation and evaluation of the first-two-char input method using a subset of the Wikipedia dataset.

4 Experimental evaluation

The experimental evaluation is divided into three main parts: splitting the dataset into training, validation, and test subsets; evaluating the performance of the methods on the validation set; and selecting the best-performing configuration for the final evaluation on the test set.

[2] https://huggingface.co/datasets/legacy-datasets/wikipedia
[3] https://dumps.wikimedia.org.

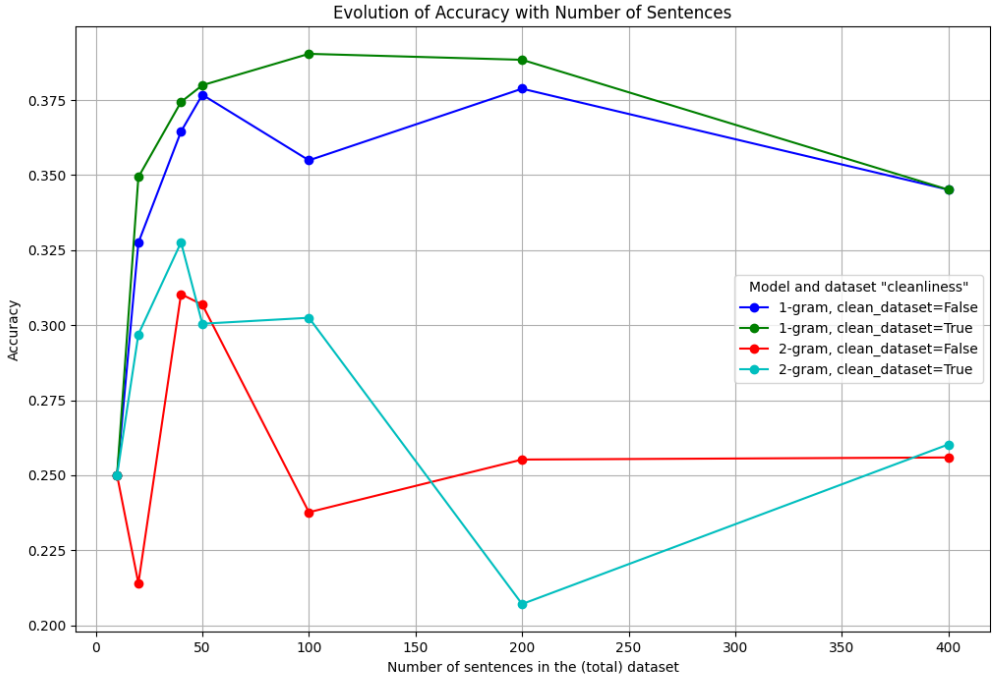

Figure 1: Evolution of the accuracy with the number of sentences.

4.1 Splitting into 3 subsets: training, validation and test'

To ensure robust evaluation of the models, the reduced dataset was split into three subsets: training, validation, and test.

We implemented a function to split the dataset into 60% training, 20% validation, and 20% test subsets, ensuring that the proportions are maintained correctly.

4.2 Evaluation of the performance of the 2 methods on the validation set in different configurations

To find the best performing configuration, we evaluated two different models (1-gram and 2-gram) and two dataset cleaning options (raw and cleaned) across various sizes of the reduced dataset. A "cleaned" dataset refers to an uncased text with punctuation removed. The results were measured in terms of accuracy, vocabulary size (number of distinct words), and execution time.

An overarching function, $prediction(input\_2\_char, model, train\_set)$, was implemented to facilitate model selection. This function prepares the data and performs predictions based on the input $input\_2\_char$ using either the 1-gram or 2-gram model based on the specified parameter.

The results of these evaluations were stored in a CSV file [4] for further analysis. We compared the accuracy, execution time, and vocabulary size of the models using different configurations. The results are visualized in Fig. 1 and 2.

When analyzing these results, it is important to keep in mind that when the sentence number changes, the sizes of the train, validation and test subsets change too. We applied each time a function to split the dataset corresponding to the specified number of sentences into 60% training, 20% validation, and 20% test subsets, ensuring that the proportions are maintained correctly. Hence, even if two configurations with different sentence numbers have the same accuracy, the one with the largest number of sentences performs better than the other one because it encounters more new and unknown words in the validation subset as this subset contains more sentences.

[4] Available on Github: https://github.com/julesdesir/First2char-input-method/blob/main/evaluation.csv

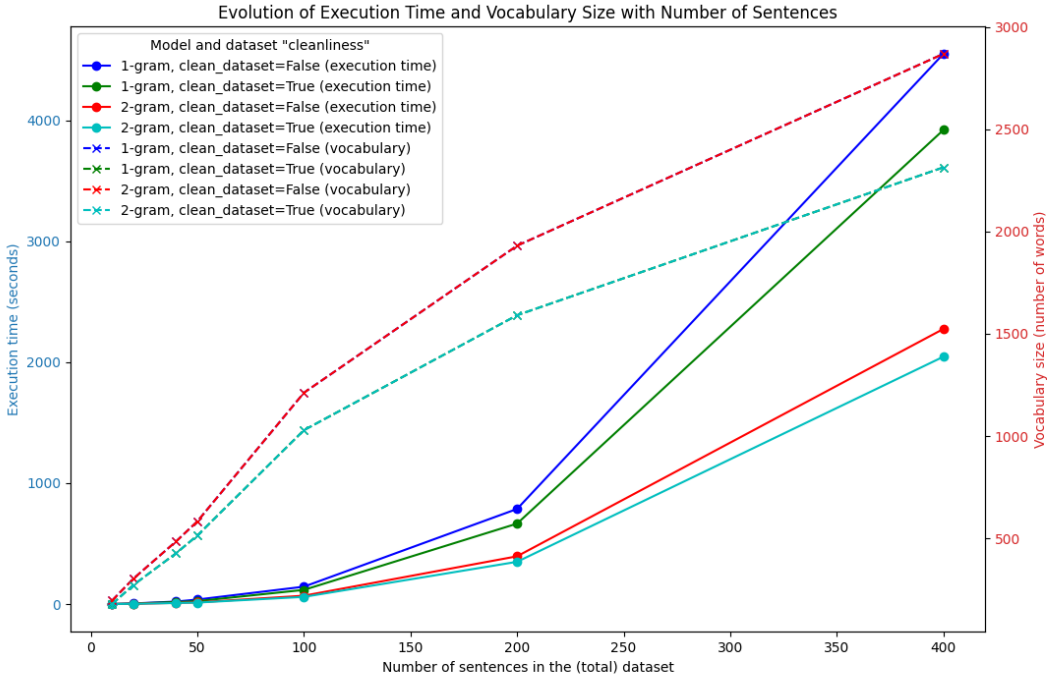

Figure 2: Evolution of the execution time and the vocabulary size with the number of sentences.

Sentence number	Model	clean_dataset	Vocabulary	Accuracy	Execution time (s)
400	1-gram	True	2314	0.36	3670

Table 1: Final results of the selected method on the test set.

4.3 Best performing configuration selection, and final evaluation on the test set

After evaluating various configurations on the validation set, we selected the best-performing combination for the final evaluation on the test set. The chosen combination was the 1-gram model with the cleaned dataset option.

This final evaluation (see Table 1) confirmed the effectiveness of the chosen configuration, providing detailed metrics on accuracy, vocabulary size, and execution time for the test set, which was not used in previous evaluations to ensure unbiased performance assessment.

5 Conclusion

This study aimed to evaluate the performance of two n-gram language models (1-gram and 2-gram) on a subset of the English Wikipedia dataset, focusing on their accuracy, vocabulary size, and execution time. Through a comprehensive experimental evaluation, the models were tested across different dataset sizes and preprocessing conditions (raw vs. cleaned text).

5.1 Key results

The key results are as follows:

- **Dataset preparation and cleaning**: Effective preprocessing, including converting text to lowercase and removing punctuation, improved the accuracy of the models. For the majority of

the dataset sizes, the cleaned datasets yielded better results compared to their raw counterparts.

- **Model performance**: Among the configurations tested, the 1-gram model with a cleaned dataset emerged as the best performer. This configuration achieved the highest accuracy while maintaining a reasonable vocabulary size and execution time. The 2-gram model, while offering more contextual information, did not outperform the 1-gram model in terms of accuracy. This low performance may be due to the fact that there were not enough training data and known bigram possibilities to make the 2-gram model perform well.
- **Execution time and vocabulary size**: As expected, increasing the size of the dataset led to a larger vocabulary and longer execution times. The 2-gram model demanded more computational resources.
- **Final evaluation**: The final evaluation on the test set was used to test the optimal configuration on a subset which remained unseen during model training and validation. This configuration delivered robust performance with an acceptable balance of accuracy, vocabulary size, and execution time.

5.2 Perspectives and future work

The results of this study underscore the importance of data preprocessing and model selection in natural language processing tasks. While n-gram models are relatively simple compared to modern deep learning approaches, they can still provide valuable insights and serve as strong baselines for more complex models.

Future work could explore the following directions:

- **Larger and more diverse datasets**: Extending the evaluation to larger and more diverse datasets would provide a more comprehensive understanding of the models' generalizability. Nevertheless, it is necessary to have more computational resources.
- **Advanced models**: Comparing traditional n-gram models with advanced deep learning models, such as transformers, could highlight the strengths and weaknesses of each approach.
- **Optimization techniques**: Investigating optimization techniques for faster execution times and smaller memory footprints could make n-gram models more viable for real-time applications. Parallelization techniques could be used for this purpose.

In conclusion, while simple n-gram models may not match the sophistication of newer techniques, they remain an important step in the NLP toolkit, especially when computational resources are limited. This study contributes to a better understanding of their capabilities and limitations, paving the way for more informed choices in language model applications.

References

[CL00] Z. Chen and K.-F. Lee. A new statistical approach to chinese pinyin input. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 241–247, 2000.

[MMYN99] S. Mori, T. Masatoshi, O. Yamaji, and M. Nagao. Kana-kanji conversion by a stochastic model. *Transactions of Information Processing Society of Japan*, 40(7):2946–2953, 1999.