International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 06 June 2024, pp: 592-597 www.ijaem.net ISSN: 2395-5252

Typing Assistant Using Deep Learning Techniques

P.Sai Vamshi, B.Sanjana, G.Sathwik, V.Shravya, D.Shreya, Prof. S.Jim Reeves

School of Engineering, Malla Reddy University

Date of Submission: 08-06-2024 Date of Acceptance: 18-06-2024

ABSTRACT: Typing Assistant is a helping tool that autocompletes writing by completing current words and suggesting relevant word sequences. A smart typing assistant is the one that can suggest more than just the current word. Based on the context, it can suggest the next few relevant words. In this work, we propose such a model using the recent advancement of the natural language toolkit . It is a context-aware model that can predict the next few words depending on the context the user is writing. This is a general- purpose solution to a typing assistant system. Our work stands out from other research in two cases. First, rather than using only a temperature variable to introduce randomness in the generation and choosing only one character, we used beam search to generate multiple-word suggestions. Second, we used encoder- decoder architecture to generate more than the one- word suggestion. Often more than one word of this suggestion is relevant enough to choose, making the typing more efficient. It is similar to an English keyboard.

I. INTRODUCTION

In today's digital era, efficient typing is crucial for communication. Traditional typing assistants often fall short, only completing single words without considering context. Our project introduces a smarter typing assistant using advanced neural networks, specifically natural language tool kit (NLTK).This model predicts not just the next word but suggests relevant word sequences based on what the user is writing.

What sets our work apart is two fold. Firstly, we use beam search to offer multiple word suggestions, ensuring better context and relevance. Secondly, our encoder-decoder architecture allows for generating coherent word sequences, making typing more intuitive and efficient.

Inspired by familiar keyboards, our goal is to provide a user-friendly solution that significantly improves typing assistance. Through our project, we aim to overcome the limitations of existing systems and open avenues for

II. LITERATURE REVIEW

1. Language modeling with the artificial neural network was proposed by Bengio el.al (2003) with a feed-forward neural network with fixed length context vector. Which was exceptionally successful bringing the focus on neural network for language modeling.
2. Later Schwenk el. al. (2005) showed that this approach is also used for speech recognition.
3. Milkolov el al. (2010) introduced a recurrent neural network for language modeling .
4. With the architecture introduced in Elman el. al. which can pass information in long sequence from front to end.

III. PROBLEM STATEMENT

"Develop a typing assistant using deep learning techniques to predict and suggest the next word or phrase as a user types on an English keyboard. The system should utilize a neural network model trained on a large corpus of text data to accurately predict the most probable next words or phrases based on the context of the input text."

This problem definition outlines the objective, the target user interface (English keyboard), and the expected functionality (predictive text suggestions using deep learning).

IV. METHODOLOGY PREPROCESSING:

The preprocessing methods used by us are as follows:

A. Counting words in Corpora:

Counting of things in NLP is based on a corpus. NLTK (Natural Language Toolkit)

International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 06 June 2024, pp: 592-597 www.ijaem.net ISSN: 2395-5252

provides a diverse set of corpora. For our project we'll be using the Brown corpus. The Brown corpus is a 1-million-word collection of samples from 500 written texts from different genres (newspaper, novels, non- fiction etc.).There are tasks such as spelling error detection, word prediction for which the location of the punctuationis important. Our application counts punctuation as words.

B. N-Grams Model:

Probabilistic models are used for computing the probability of an entire sentence or for giving a probabilistic prediction of what the next word will be in a sequence. This model involves looking at the conditional probability of a word given the previous words.

If we consider each word occurring in its correct location as an independent event, we might represent this probability as follows:

$$P(w_1, w_2 \ldots, w_{n-1}, w_n)$$

We can use the chain rule of probability to decompose this probability:

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1})$$
$$= \prod_{k=1}^{n} P(w_k\, w_1^{k-1})$$

C. Bigram Model:

In this model we approximate the probability of a word given all the previous words by the conditional probability of the preceding word.

$$P(w_n|w_1^{n-1}) = P(w_n|w_{n-1})$$

For a bigram grammar, then, we compute the probability of a complete string:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

To calculate the probability, from this corpus we take the count of a particular bigram, and divide this count by the sum of all the bigrams that share the same first word.

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

D. Trigram Model:

A trigram model looks just the same as a bigram model, except that we condition on the two-previous words.

E. Minimum Edit Distance:

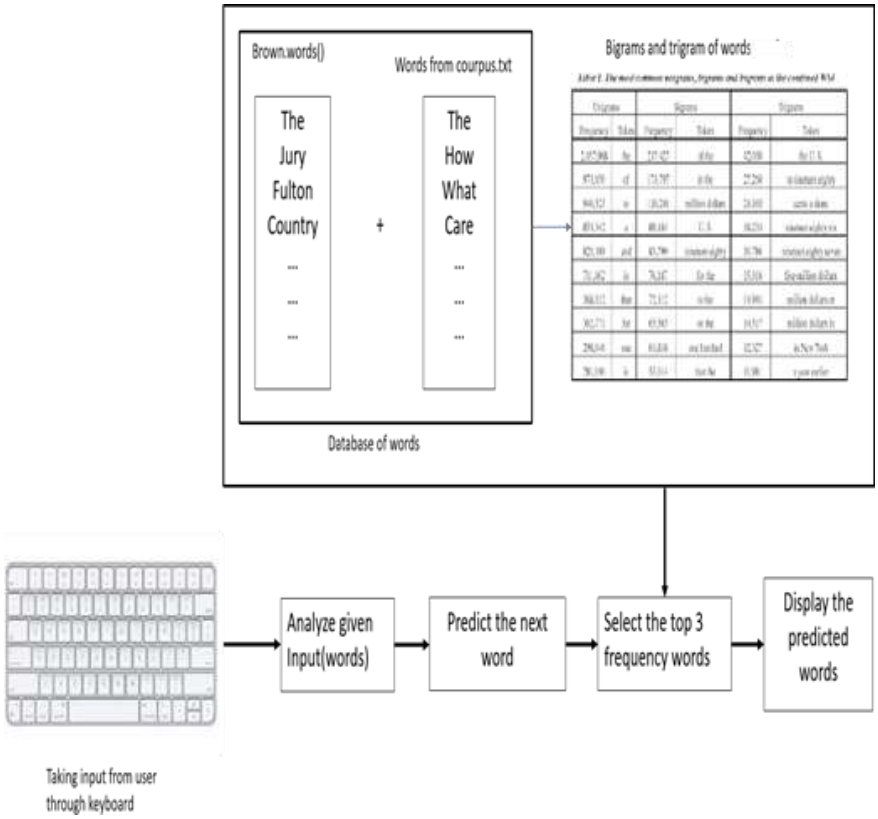
The distance between two strings is a measure of how alike two strings are to each other. The minimum edit distance between two strings is the minimum number of editing operations (insertion, deletion, substitution) needed to transform one string into another.

$$P(t|c) = \min \begin{cases} distance[i-1, j] + ins\text{-}cost(target_j) \\ distance[i-1, j-1] + subst\text{-}cost(source_j, target_i) \\ distance[i, j-1] + ins\text{-}cost(source_j)) \end{cases}$$

Minimum edit distance is used in the correction of spelling mistakes or OCR errors, and approximatestring matching, where the objective is to find matches for short strings in many longer texts.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 06 June 2024, pp: 592-597 www.ijaem.net ISSN: 2395-5252

ARCHITECTURE



MODEL DEVELOPMENT:

Using Bigram and Trigram model to suggest predictions on the software keyboard:

To predict words for a given a sequence, a bigram and a trigram module were implemented in python. The bigram module calculates the probability of occurrence of a word after a given string. This is achieved by storing all the possible words in the corpus which can follow a given previous word and thecount of this bigram as a key-value pair in a hash map. The probability can be calculated by dividing thevalue (count) by the number of times the given word occurs in the corpus.

Similarly, the trigram module is stored as a hash map of hash maps consisting of the possible words following a sequence of words (two words) with their respective counts. Hash maps are used to achievefaster lookups.

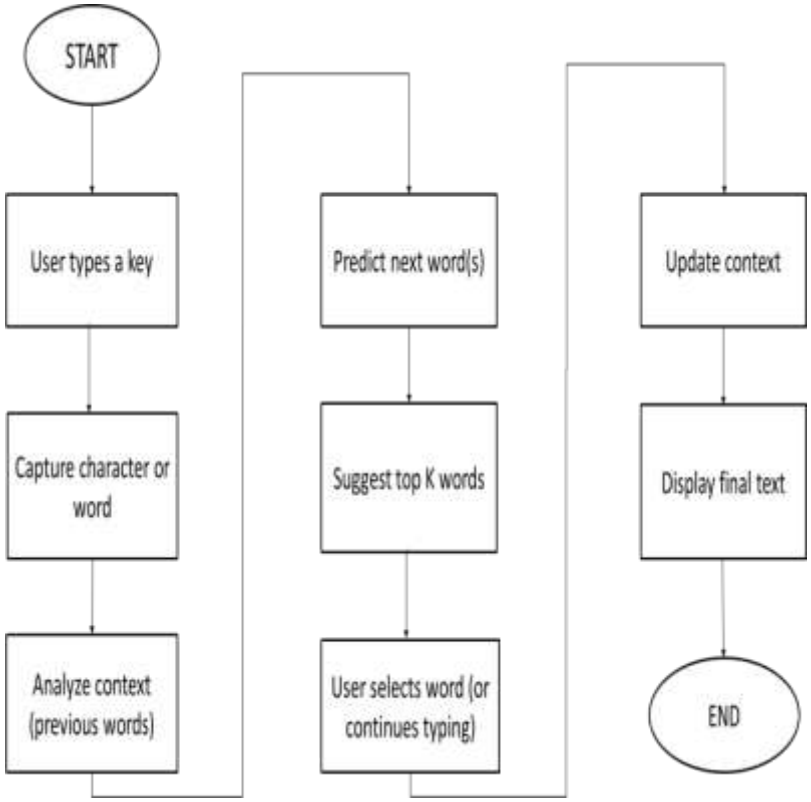
Using Minimum Edit Distance Module for auto-completion:

Often in real world typing, a user may mistype a word, which an intelligent typing assistant should be able to suggest corrections for. This is implemented using the concept of Minimum Edit Distance between predictions and what the user typed. We implement this using dynamic programming that findsthe minimum number of additions, subtractions, or substitutions required to covert one word to another. However, when using this module with the large number of words in the corpus, we realise that it is not very time-efficient.

Hence inour final implementation, we use the nltk function for finding theLevenshtein distance between given words. We also allow unit transposition cost to factor in cases wherethe user may have typed "fisrt" instead of "first", as is common in fast typing

International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 06 June 2024, pp: 592-597 www.ijaem.net ISSN: 2395-5252

V. MODEL IMPLEMENTATION



VI. RESULTS

International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 06 June 2024, pp: 592-597 www.ijaem.net ISSN: 2395-5252

VII. CONCLUSION

In summary, our model for a context-aware typing assistant, powered by Natural language toolkit(NLTK) with encoder-decoder architecture and beam search, offers an effective solution to enhance word suggestion efficiency during typing. By generating multi-word suggestions aligned with user context, we've overcome the limitations of existing systems, making typing more intuitive and efficient. Further improvements could refine the model, but overall, our approach represents a significant step forward in improving written communication.

VIII. FUTURE WORK

The future scope of a typing assistant using deep learning, particularly with NLTK (Natural Language Toolkit), and an English keyboard is promising. With ongoing advancements in deep learning, the typing assistant could become more intuitive, adaptive, and capable of understanding context and user intent better, leading to improved accuracy and efficiency in typing assistance. Additionally, integration with other technologies like natural language processing (NLP) and machine translation could further enhance its capabilities, making it more versatile across different languages and communication contexts.

REFERENCES

[1]. Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural proba-bilistic language model," Journal of machine learning research, vol. 3,no. Feb,pp. 1137–1155, 2003.

[2]. H. Schwenk and J.-L. Gauvain, "Training neural network language models on very large corpora," in Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Lan-guage Processing.

Association for Computational Linguistics, 2005,pp. 201–208.

[3]. T. Mikolov, M. Karafi´at, L. Burget, J. ˇCernock`y, and S. Khudanpur,"Recurrent neural network based language model," in Eleventh Annual Conference of the International Speech Communication Association,2010.

[4]. J. L. Elman, "Finding structure in time," Cognitive science, vol. 14,no. 2, pp. 179–211, 1990.

[5]. M. Boden, "A guide to recurrent neural networks and backpropagation ,"the Dallas project, 2002