Text Prediction using Probability Matrices and Prefix Trees

John Alvin L. Sayson

I. Introduction

Verbal communication was always been a vital component of human interaction. After all, it allows two people to be able to understand each other. With the influx and rapid growth of technology over the previous decades, new methods of communication in the digital age were conceived over that timespan.

Nowadays, prediction keyboards are commonplace in mobile phones. Numerous mobile operating systems seamlessly integrate prediction into the existing keyboard functions to the point where people use it switching from the regular keyboard, to the prediction keyboard, and back again.

It might be common in mobile systems, but what about desktop ones? This study involved investigation on two online applications that provide generated outputs from predefined inputs.

AIWriter [1] and Botnik's Voicebox are these two applications. AIWriter's process is fully automatic, instantly generating works depending on settings provided by the user in the beginning. On the other hand, Voicebox is more of a hands-on process as the user selects the words suggested by the application in writing their literary works. The suggested words are determined by the source text the user selected in the beginning, ranging from Harry Potter narrations to pancake recipes. If the selection range does not interest the user, they can provide their own source texts by uploading it to the website.

Both website applications improve writing time by doing away with manual word-for-word typing, providing either words or complete paragraphs for the users to use right away. This study intends to provide a similar convenience.

Writing or typing time varies from person to person. For those with cognitive, perceptive, and/or physical disabilities, predictive text systems were their preferred medium in being able to communicate easier in social settings. [2] Being able to just use one tap to select words rather than multiple taps per letter, it allows for a great increase in convenience in conversation. As said before, predictive text systems are now embedded in various native keyboards in mobile phones, wherein continuous usage can further personalize the suggested words to the user. Through this, the rate of communication between people can improve. This application aims to achieve a similar goal to that of the common predictive text system. By providing an interface wherein users are given suggestions while in the middle of creating texts, they are able to utilize the power of mobile prediction keyboard functionalities.

The application that was implemented as the main output of this study differs from the aforementioned Voicebox is the way the initial input is accepted. The application accepts user input as the initial data for the probability matrix and prefix tree, which are the two main data structures used for the application.

This paper shall discuss the predictive text application in detail, as well as its implementation.

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

II. Objectives

A. General Objective

The study aims to create an application that utilizes similar functionalities as a mobile prediction keyboard as an extension for the Google Chrome browser.

B. Specific Objectives

1. Receive initial input from the user to instantiate the predictor application;
2. Use the probability matrix data structure in mapping word-level relation;
3. Use the prefix tree data structure in word prediction;
4. Present word- and letter-level relation to the end user in the form of a predictive text writer interface;
5. Allow user text selection to create texts and allow said texts to update previously mentioned data structures; and
6. Assess user experience and program outputs using surveys.

III. Review of Related Literature

To understand the possible applications and previous implementations of story and text generation, previous works in the field were observed.

Uchimoto, et al. presented various text-generation models in their paper wherein Japanese sentences are formed by feeding the program keywords which serve as the subjects of the sentence it produces. [3]

A table of input-output pairs were provided by the authors, with inputs being words in groups of threes and the outputs being complete sentences. From this, a possible limitation provided by their application was that it could only accept three inputs at a time.

The authors have mentioned that the program was created for speakers that are not as fluent in Japanese be able to

express sentences that they intend to speak but only know the main words that they want to express. This shows that text generation is not limited to recreational applications, and can be used for accessibility and elaboration.

X. Zhang and M. Lapata utilized recurrent neural networks in 2014 to generate Chinese poetry, which follows a specific format for it to be considered one. [4] The Chinese poems that they generated follow the quatrain format, wherein four lines of poetry must have five or seven characters each. The poetry generator functions similarly to Uchimoto's 2002 paper where keywords are accepted to outline the poem's main concept. In creating the first line, a language model is used to rank candidates that satisfies criteria defined by the poetry format. Following lines are then based on the previous lines. This will ensure continuity from the first line to the last.

Text generation has also ventured into literary fields with less constraints, specifically story writing.

In 2018, A. Fan, et al. added sentence prompt generation (aside from the expected story generation component) which was then based on to create the corresponding stories. [5] These prompts are created using a convolutional language model, which contains a novel gated self-attention mechanism created by some of the authors. [6] They also used a sequence-to-sequence network, a model that uses two recurrent neural networks as encoders and decoders, to generate the story based on the prompts trained on the convolutional language model. The sequence-to-sequence model depended upon the trained prompts from the convolutional language model, resulting in a fusion model, as referred to by the writers.

Results shown in the paper contained comparisons between the fusion model and another language model, presenting the former's readability and higher quality of output over the latter. However, the authors mentioned that one of the limitations of the fusion model is the genericness of the prompts generated, compared to human prompts. This limits the results that can be generated as more specific and varied prompts can provide better results.

IV. Methodology

This section contains the features of the implementation of the predictive text application, as well as the evaluation procedures that follow thereafter.

A. Application Implementation

The application was implemented using only JavaScript, along with the Chrome Extension API to integrate the application with the browser.

B. Text Input Representations

1) Sequence: The sequence is a simple array of strings from the input text, split by a single space (" ") delimiter.

2) Bag-of-Words: The bag-of-words then uses this sequence to count word occurrences.

3) Probability Matrix: The probability matrix uses both previously mentioned representations to produce the probability of a word to appear given another word. This is represented as the matrix keys, which is obtained from the keys of the bag-of-words.

4) Prefix Tree: The prefix tree uses the sequence representation to create the network of letters.

C. Application Specifications

1) Input: Upon installation of the extension, preliminary input data is accepted in a HTML page where the user is asked to enter it.

2) Processing: Upon selection of the [Begin Predicting] button in the page for the newly-installed extension, these words are then converted into the four representations mentioned earlier.

The created matrix and prefix tree is stored on the local storage of the browser. The bag-of-words and sequence representations are stored inside the matrix as reference.

Before producing these representations of the text, the text is first cleaned in order to remove various characters and long spaces, which is vital for creating the sequence.

When the user clicks on the [Copy to Clipboard] button on the Text Entry page, the new input is converted into the four representations like in the page for the newly-installed extension. However, instead of directly storing it in the local storage, the application retrieves the saved matrix and prefix tree and merges it with the new ones.

3) Output: While typing in the available text box in the Text Entry Page, the caret location in said text box is used in determining the level of prediction for the next word. If the previously typed word has a space after it, it performs word-level prediction. Else, it performs letter-level prediction.

Word-level prediction uses the matrix, which is why it was important to use the words that appeared in the inputs as the keys. Letter-level prediction uses the prefix tree.

Determining results from the letter-level prediction is just navigating the tree for each letter of the word to be searched and then adding all results from that point onward into an array.

For word-level prediction, the class NameProbabilityPair is used in arranging the most probable words by percentage. Determining the values for the NameProbabilityPair is defined using the equation from the Bayes' theorem, where A is the word to generate possible outputs from, and B is one of the candidate words:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

Meanwhile, the probability of a word is given by this equation:

$$P(A) = \frac{count(A)}{count(total)}$$

The product of the two are then divided by the probability of B, obtained the same way as A.

The results from these predictions are displayed on the area below the text box, the number which is defined by the user in Settings (default is 6).

V. Results and Discussion

This section shall discuss the resulting interface, as well as the evaluation results.

A. Interface

The application consists of three main screens, the New Predictor screen, the Text Entry screen, and the Settings screen.

1) New Predictor: This screen appears when the extension is installed or upon reset of the predictor.

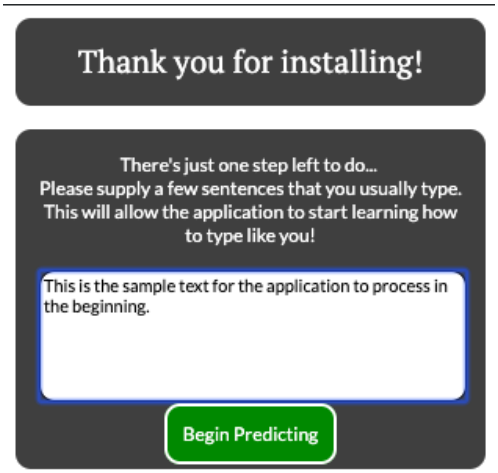

Fig. 1. The New Predictor screen with input posted.

2) Text Entry: This screen appears when the extension is installed. This also appears upon reset of the predictor.

Fig. 2. The Text Entry screen with posted results based off of the input in the previous screen.

As mentioned before, the number of words that appear in the option below can be modified by the user in Settings. Fig. 3 illustrates that functionality, as well as the word-level prediction for the previous input, which is the word ”this”.

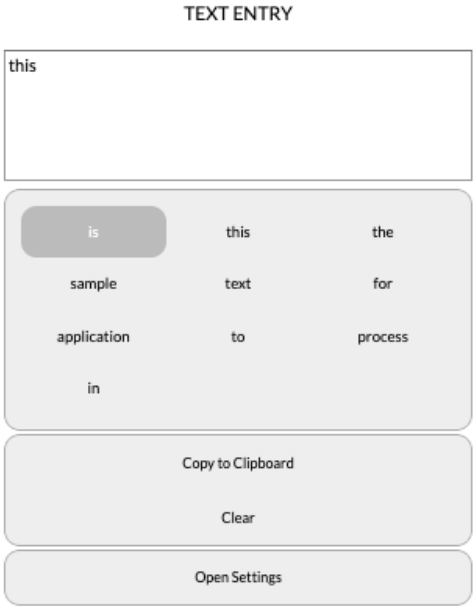

Fig. 3. The Text Entry screen with more keywords.

Fig. 4 showcases the letter-level prediction functionality in the application with the input ”ap”. Do note that the other words were placed manually before this screenshot.

Fig. 4. Letter-level prediction functionality.

3) Settings: Clicking the Open Settings button loads this screen. This contains the setting of the number of keywords, as well as the option to reset the application to its default settings with an empty matrix, prefix tree, etc.

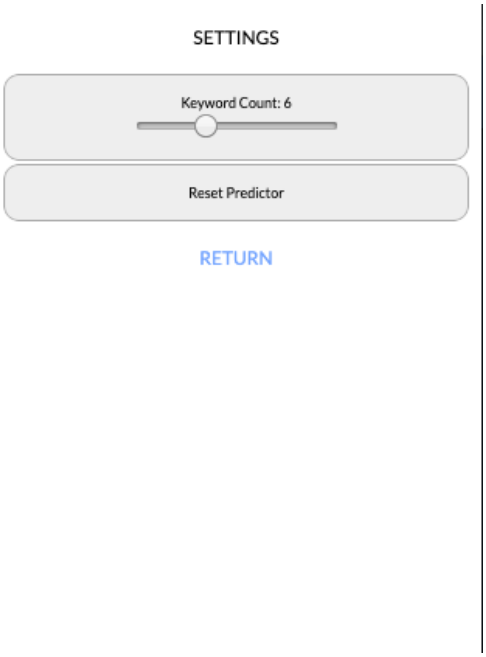

Fig. 5. The Settings screen.

B. Application Evaluation

The application was evaluated by thirty respondents, not taking into account any personal information, just their experience with the application.

The survey questions asked for the user's evaluation about the following:

1. Accessibility of the application
2. Word-level prediction accuracy
3. Letter-level prediction accuracy
4. Application robustness
5. Overall user experience
6. Overall application rating

Here are the gathered results from the survey.

1) Accessibility of the Application: The input type for evaluating the accessibility of the application results was a linear range from 1 to 5, where 1 means that the application is not accessible at all while 5 means very accessible. The resulting average of the user's inputs result to 4.2667, which suggests that the application is generally easy to install and accessible to users involved.

2) Word-level Prediction Accuracy: The input type for evaluating the word-level prediction accuracy was also a linear range from 1 to 5, where 1 means that the word-level prediction is not accurate at all or it produces a wrong output while 5 means that it is very accurate. The resulting average of the user's inputs result to 4.2333, which suggests that the application performs word-level prediction with an above-average accuracy.

3) Letter-level Prediction Accuracy: The input type for evaluating the letter-level prediction accuracy was also a linear range from 1 to 5, where 1 means that the letter-level prediction is not accurate at all or it produces a wrong output while 5 means that it is very accurate. The resulting average of the user's inputs result to 4.4, which suggests that the application performs letter-level prediction with an above-average accuracy. However, it is of note that there is one instance of the letter-level prediction being inaccurate, leading to a single 1 response for this section.

4) Application Robustness: The input type for evaluating the application robustness was a multiple choice with the options being Yes or No. A 96.7 percentage of Yes responses suggest that the application generally works most of the time. It is worth noting that the person who responded with a No did not provide an error for the follow-up question.

5) Overall User Experience: The input type for evaluating the overall user experience was also a linear range from 1 to 5, where 1 means that the overall user experience was not intuitive for users while 5 suggests the opposite. The resulting average of the user's inputs result to 3.9, which means that the application does provide an interface that is intuitive enough but needs room for improvement.

6) Overall Application Rating: The input type for evaluating the overall application rating was also a linear range from 1 to 5, where 5 means that the application performs functionalities properly and correctly while 1 suggests the opposite. The resulting average of the user's inputs result to 4.2, which means that the application generally performs its functions without much issue.

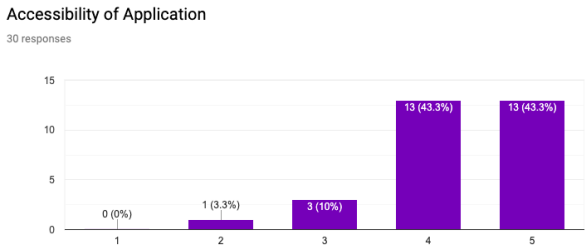

Fig. 6. Accessibility of application results

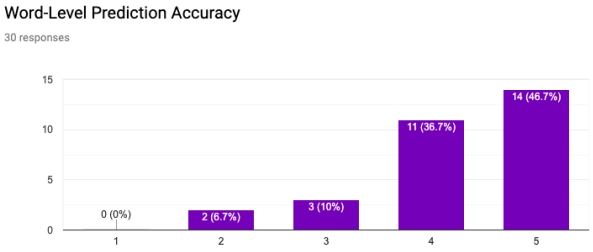

Fig. 7. Word-level prediction accuracy results

VI. Conclusion and Future Work

The application produced was able to perform predictions both in the letter-level and the word-level, implemented using

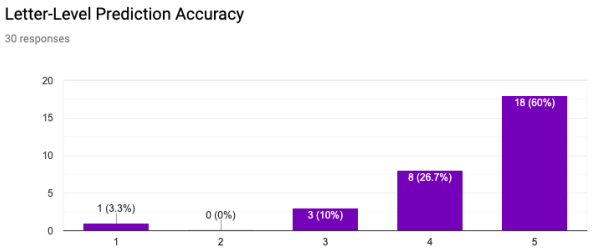

Fig. 8. Letter-level prediction accuracy results

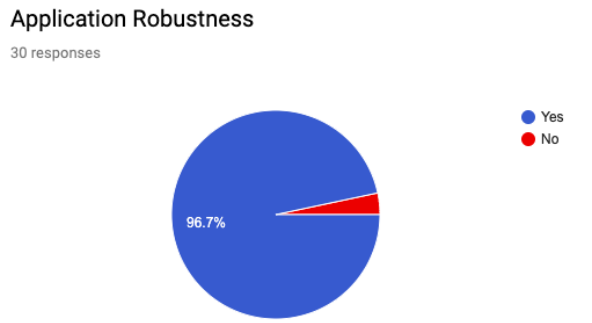

Fig. 9. Application robustness results

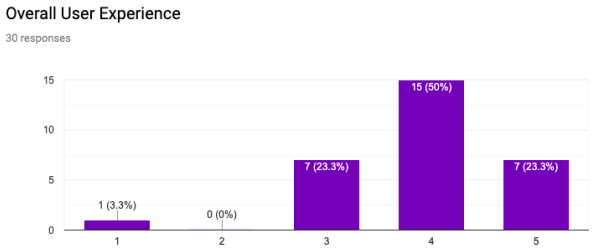

Fig. 10. Overall user experience results

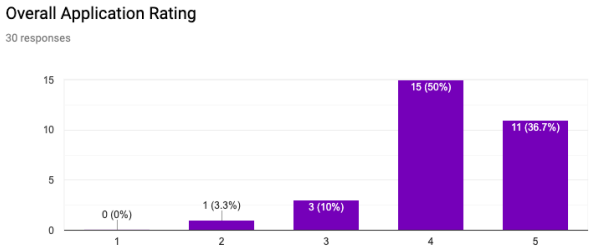

Fig. 11. Overall application rating results

four representations of the text input. Upon evaluation, the application performs favorably under most conditions.

However, during the evaluation through surveys, the comments stated that the user interface needed some work regarding text sizes and design consistency. Considering as the overall user experience received the least evaluation of 3.9, further enhancements to this component should be observed. Some comments also suggested the possibility of preloading a huge dataset of words into the prefix tree in order.

Some comments also considered using this kind of application for making predictions using tweets as an input, which would involve data mining for scraping tweets.

REFERENCES

[1] "Aiwriter," http://www.ai-writer.com/, accessed: 2018-11-01.
[2] N. "Garay-Vitoria and J. Abascal, ""text prediction systems: a survey"," *"Universal Access in the Information Society"*, vol. "4", no. "3", pp. "188–203", "Mar" "2006". [Online]. Available: "https://doi.org/10.1007/s10209-005-0005-9"
[3] K. Uchimoto, H. Isahara, and S. Sekine, "Text generation from keywords," in *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, ser. COLING '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 1–7. [Online]. Available: https://doi.org/10.3115/1072228.1072292
[4] X. Zhang and M. Lapata, "Chinese poetry generation with recurrent neural networks," pp. 670–680, 01 2014.
[5] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," *CoRR*, vol. abs/1805.04833, 2018. [Online]. Available: http://arxiv.org/abs/1805.04833
[6] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," *CoRR*, vol. abs/1612.08083, 2016. [Online]. Available: http://arxiv.org/abs/1612.08083
[7] "Voicebox," https://botnik.org/apps/writer/, accessed: 2018-11-01.

John Alvin L. Sayson is an undergraduate student of the University of the Philippines Los Baños. He likes to spend the day sleeping, playing games and occasionally playing music with his keyboard.