TABLE OF CONTENTS

Abstract

With growth in mobile technologies and vast spread of the internet, socializing has become much easier. People around the world spend more and more time on their mobile devices for email, social networking, banking and a variety of other activities. Due to fast paced nature of such conversation, it is necessary to save as much as time possible while typing. Hence a predictive text application is necessary for this. Text prediction is one of the most commonly used techniques for increasing the rate of communication. However, the speed at which text is predicted is also very important in this case. The objective of this work is to design and implement a new word predictor algorithm that suggests words that are grammatically more appropriate, with a lower load for system and significantly reduce the number of keystrokes required by users. The predictor uses a probabilistic language model based on the methodology of the N-Grams for text prediction.

Project Overview

The goal is to replicate a similar behavior of human word selection into a natural language processing model. Suggestions of new words that might be used are generated based upon the previous set of words. To achieve this goal, the N-Grams model is used. We assign probability to each word and select the next word with highest probability values. This probability values are generated based upon the sequence of previous words. These sequences are known as N-Grams.

An important aspect which needs to be understood is the speed at which the words are predicted. Since one of the popular applications of text prediction is in mobile communication, it is necessary that the load on the system as well as the speed of prediction should be optimal. This requires some tradeoff between the accuracy of the predicted words and speed. Project is based on word prediction on sentence by using stochastic, i.e., N-gram language model such as unigram, bigram, trigram, minimum edit distance and backoff models for auto completing a sentence by predicting a correct word in a sentence which saves time and keystrokes of typing and also reduces misspelling.

Related Work

1. An Approach for a Next-Word Prediction for Ukrainian Language

A personalized text prediction system is a vital analysis topic for all languages, primarily for Ukrainian, because of limited support for the Ukrainian language tools. LSTM and Markov chains and their hybrid were chosen for next-word prediction. Their sequential nature (current output depends on previous) helps to successfully cope with the next-word prediction task. The Markov chains presented the fastest and adequate results. The hybrid model presents adequate results but it works slowly. Using the model, user can generate not only one word but also a few or a sentence or several sentences.

2. A RNN based Approach for next word prediction in Assamese Phonetic Transcription

Here, a Long Short Term Memory network (LSTM) model which is a special kind of Recurrent Neural Network (RNN) for instant messaging, where the goal is to predict next word(s) given a set of current words to the user. This method is more complex in other languages apart from English. For instance, in Assamese language, there are some equivalent synonyms of the word 'you', that is used to address a second person in English. Here, they have developed a solution to this issue by storing the transcripted Assamese language according to International Phonetic Association (IPA) chart and have fed the data into our model. Our model goes through the data set of the transcripted Assamese words and predicts the next word using LSTM with an accuracy of 88.20% for Assamese text and 72.10% for phonetically transcript Assamese language. This model can be used in predicting next word of Assamese language, especially at the time of phonetic typing.

3. Next Word Prediction Using Hindi Language

Natural language generation is a process that concerns on generating human understandable language. This study provides method to guess next word from previous sequence of words that are in Hindi language. This process reduces the keystrokes of a user by predicting next word. In this two-machine learning technology, BERT (Bidirectional Encoder Representations from Transformers) Model and ML (Masked Language) model is used to predict next word from previous words. Next word prediction is a technology that takes input and simplify typing process as of suggest user the next word according to the understanding of previous words. It makes typing less time consuming and error free. In current times, everyone is writing data digitally and to make typing efficient in Hindi language we require such systems. There are various systems related to English Language. Similar systems are required for Hindi language as Hindi is a more frequent language in India.

Methodology

Predicting the next word has been an important technique for better communication for more than a decade. Traditional systems used word frequency lists to complete the words already spelled out by the user. The implementation involves using a large corpus. The methods used are as follows:

1. Counting words in Corpora:

Counting of things in NLP is based on a corpus. NLTK (Natural Language Toolkit) 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 punctuation is important. Our application counts punctuation as words.

2. N-Grams Models:

Let's begin with the task of computing P(w|h), the probability of a word w given some history h. Suppose the history h is "its water is so transparent that" and we want to know the probability that the next word is the:

$$P(\text{the}|\text{its water is so transparent that}). \quad (3.1)$$

One way to estimate this probability is from relative frequency counts: take a very large corpus, count the number of times we see its water is so transparent that, and count the number of times this is followed by the. This would be answering the question "Out of the times we saw the history h, how many times was it followed by the word w", as follows:

$$P(\text{the}|\text{its water is so transparent that}) = \frac{C(\text{its water is so transparent that the})}{C(\text{its water is so transparent that})} \quad (3.2)$$

With a large enough corpus, such as the web, we can compute these counts and estimate the probability from Eq. 3.2.

we wanted to know the joint probability of an entire sequence of words like its water is so transparent, we could do it by asking "out of all possible sequences of five words, how many of them are its water is so transparent?" We would have to get the count of its water is so transparent and divide by the sum of the counts of all possible five-word sequences. Now how can we compute probabilities of entire sequences like P(w1,w2,...,wn)? One thing we can do is decompose this probability using the chain rule of probability:

$$\begin{aligned} P(X_1 \ldots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_{1:2}) \ldots P(X_n|X_{1:n-1}) \\ &= \prod_{k=1}^{n} P(X_k|X_{1:k-1}) \end{aligned} \quad (3.3)$$

Applying the chain rule to words, we get

$$\begin{aligned} P(w_{1:n}) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \ldots P(w_n|w_{1:n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_{1:k-1}) \end{aligned} \quad (3.4)$$

The chain rule shows the link between computing the joint probability of a sequence and computing the conditional probability of a word given previous words. Equation 3.4 suggests that we could estimate the joint probability of an entire sequence of words by multiplying together a number of conditional probabilities.

- Bigram Model:
  In this model the probability of a word is approximated by given all the previous words by the conditional probability of the preceding word. For a bigram, we compute the probability of a complete string. 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.

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

3. Minimum Edit distance:

The distance between two string 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 approximate string matching, where the objective is to find matches for short strings in many longer texts.

Proposed Solution Architecture

A. Cleaning the Data Sources

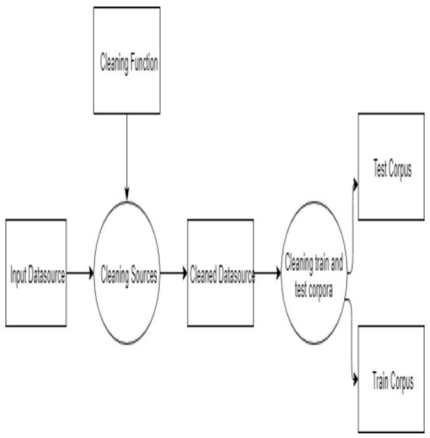


Figure 1: Cleaning of data sources and creating Test and Train corpus
(Source: https://www.ijitee.org/wp-content/uploads/papers/v8i5/E3050038519.pdf)

Figure 1 shows the process of cleaning the data sources and generating the training and testing models. Cleaning function cleans a source of text (blogs, news, or twitter) using standard and adhoc basic cleaning functions. The standard cleaning functions came from R base and the 'tm' package. The result is a normalized version of the source, keeping its structure. Table 2 summaries the list of functions applied.

B. Generating Prediction Tables

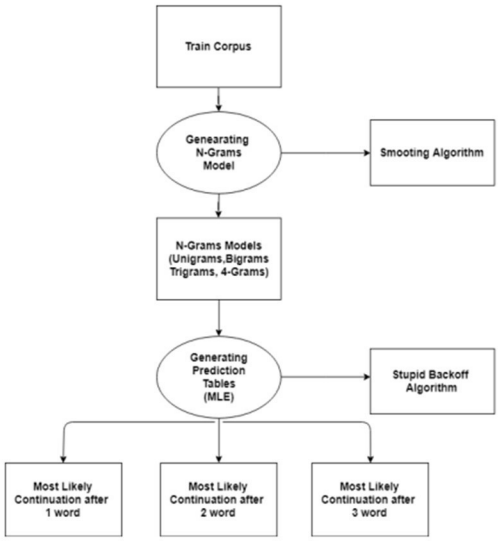


Figure 2: Generating prediction tables by applying N-Grams model on the training corpus
(Source: https://www.ijitee.org/wp-content/uploads/papers/v8i5/E3050038519.pdf)

For a n gram x that occurs r times, the probability of Good Turing is given by

$$P(x) = \frac{(r+1)(Nr+1)}{(N \cdot Nr)}$$

After smoothing, prediction tables are generated. The prediction table contains the probability of words and it helps to determine which words is most likely to appear next. It always looks for the best next words table. Stupid back off algorithm is used for handling out of vocabulary sequences. In this algorithm if a sequence is not present in a higher-order N-grams, we remove the first word and back off to a lower-order N-gram, weighing the result by a fixed factor 'lambda'.

C. Predicting Next Word and Estimating Accuracy

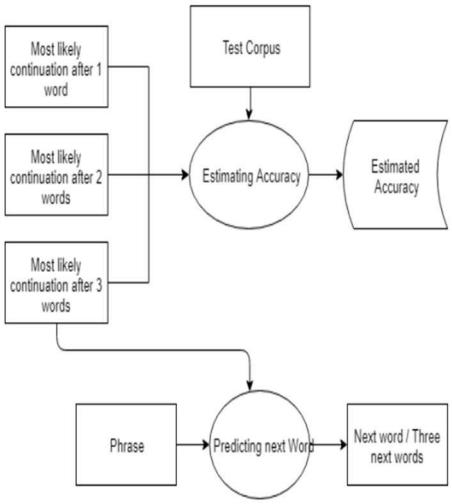


Figure 3: Predicting the next word and estimating the accuracy

(Source: https://www.ijitee.org/wp-content/uploads/papers/v8i5/E3050038519.pdf)

This module is responsible for evaluating the performance of a n gram model against an unseen corpus of text. For this module we give input the testing dataset as an unseen corpus of text. Accuracy as a ratio of correct predictions over total predictions is estimated from each n gram order. Two options are evaluated: a prediction is right if it matches only the first next word predicted or it is right if it is in the three next words proposed.

Experimentation and Results

Prediction

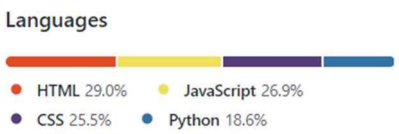
The prediction algorithm takes the entered text, cleans and extracts the preceding 1 to n-1 words. It then uses a simple backoff approach in combination with weighting to build a list of probable next words.

- For each n, select the top 3 matches based on n-gram frequency;
- Combine the results;
- Sort by maximum likelihood (high to low);
- The top items form the prediction

Accuracy

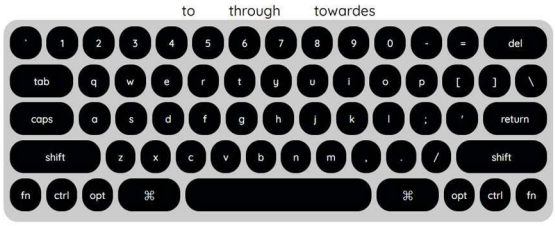
The lines of text in the dataset that were not included in the training set are used to test the model. The test selects lines at random, and splits each sentence in n-grams. The next word prediction accuracy is tested on each (n-1)-gram. The predicted word is then compared with word n. The accuracy is the ratio of correct predictions and total predictions. The test is done for the different training set sample sizes. The model accuracy varies between 8 percent and 18 percent. Varying the training set sample size between 1 and 10 percent does not seem to have a significant effect on prediction accuracy. It is to be investigated whether a larger sample size will increase accuracy.

The screenshots of model are as follow:



I am not g

I am not going t



I am not going to

I am not going to get up

Shift, opt are highlighted while taking the snapshots

Conclusion and Future Scope

This model shows that N-Grams being a fairly simple language model can prove to be efficient for real time prediction requirements. The accuracy of the model can be seen improving as the value of NGrams increases. Consider further improvements: building a 5-grams model avoiding pruning when while training the models or applying more powerful smoothing algorithms than 'stupid backoff' can be applied for increase the accuracy of the model. However, as the size of model increase the prediction time would also increase. Hence the tradeoff between accuracy and speed needs to be considered while enhancing the model.

This work towards next word prediction in phonetically transcripted Assamese language using LSTM is presented as a method to analyze and pursue time management in e-communication. We compared the accuracy of our RNN model between Assamese language and Phonetically transcripted Assamese language. From the results, it is seen that first data-set (Assamese text without transcription) have more accuracy compared to the accuracy of the second dataset (Assamese text with transcription). This is due to an increase in the number of words in the second data-set being phonetically transcripted. It will be

interesting to observe the results with much bigger datasets than the currently used one. Preparation of a big dataset of phonetically transcripted local Indian languages is a tedious job as no online sources are available for a lot of them and it has to be processed manually. Since a database of phonetically transcripted words in the Assamese language is not available, we tried to increase the number of words by repetition. In our future course of action, we plan to build larger datasets than the currently used one, implement our models and analyze the results. Even though the Assamese language is focused in our work, this LSTM model can be applied to other local languages also.

References

1. Horstmann Koester, H. & Levine, S. (1996). Effect of a word prediction feature on user performance. Augmentative and Alternative Communication, 12, 155-168.

2. Morris, C., Newell, A., Booth, L., Ricketts, I., & Arnott, J. (1992). Syntax PAL: A system to improve the written syntax of languageimpaired users. Assistive Technology, 4, 51-59.

3. Korvemaker and R. Greiner. 2000. Predicting Unix command lines: adjusting to user patterns. In Proceedings of the National Conference on Artificial Intelligence.

4. Jacobs and H. Blockeel. 2001. The learning shell:automated macro induction. In Proceedings of the International Conference on User Modelling.

5. Shannon, C. E. 1951. Prediction and entropy of printed English. Bell System Technical Journal, 30:50–64.

6. Talbot, D. and M. Osborne. 2007. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. EMNLP/CoNLL .

7. Rashid T.A., Mustafa A.M., Saeed A.M. (2018). Automatic Kurdish Text Classification Using KDC 4007 Dataset. In: Barolli L., Zhang M., Wang X. (eds) Advances in Internetworking, Data & Web Technologies.
EIDWT 2017. Lecture Notes on Data Engineering and Communications Technologies, vol 6. Springer, Cham. Doi: https://doi.org/10.1007/978-3-319-59463- 7_19

8. Thackston Wheeler (2006). Sorani Kurdish: A Reference Grammar with Selected Readings. Archived from the original on 2017-06-07. Retrieved 2017-06-06.