

A MINI PROJECT REPORT

On

TEXT PREDICTOR USING N-GRAMS

Submitted in partial fulfillment of the requirement of University of Mumbai for the Course

**In**

# Computer Engineering (IV SEM)

Submitted By

**Janhavi Anap**

**Riddhi Narkar**

Subject Incharge

**Prof. Merlin Priya Jacob**



**CERTIFICATE**

This is to certify that the requirements for the project report entitled ‘**Text Predictor using n-grams ’** have been successfully completed by the following students:

|  |  |
| --- | --- |
| **Name** | **Moodle Id** |
| Janhavi Anap | 19102043 |
| Riddhi Narkar | 19102003 |
|  |  |

In partial fulfilment of the course Python Programming (CSL405) in Sem: IV of Mumbai University in the Department of Computer Engineering during academic year 2020-2021.

Sub-in-Charge

# 

# PROJECT APPROVAL

The project entitled ‘**Text Predictor using n-grams**’ by **Janhavi Anap**, and **Riddhi Narkar** are approved for the course of Python Programming ( CSL405 ) in Sem: IV of Mumbai University in the Department of Computer Engineering.

Subject-in-Charge

Date:

Place: Thane

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr**. **No.** | **Topic** | **Page No.** |
| 1. | Abstract (150-200 words) | 5 |
| 2. | List of Figures | 6 |
| 3. | List of Tables | 6 |
| 4. | Problem Definition | 7 |
| 5. | Introduction | 8 |
| 6. | Description of the modules used |  |
| 7. | Implementation details with screen-shots (stepwise) |  |
| 8. | Conclusion and Future Scope |  |
| 9. | References |  |
| 10. | Acknowledgement |  |

**Abstract**

One of the major technological marvels of this tech-infused 21st century is the ability of us to connect digitally. With the advent of sophisticated communication tools, being able to express oneself through a digital medium effectively and efficiently needed a reliable mechanism. A very popular method of expressing thoughts, ideas or sharing information, is texting a message or posting a typed text piece on the internet. Often, typing can be prone to errors and not much efficient and quick when it comes to long typing sessions.

A word predictor is a piece of software which suggests words after processing your input. The input here could be a string of any number of words, and it normally depends on the algorithm used. An algorithm could be processing in a static way, which doesn’t learn, no matter how much it is trained and tested, or it can be dynamic, and thus, adapting the changes in diction, grammar and style of the user. A dynamic word predictor may use a corpus or a database that changes every time a user uses the service and thus updates itself with the changes and advancements of the user. Also, such a system can very easily survive the changing language style, new words, new phrases, or new slangs and hence is versatile and needs less maintenance. In addition to texting and using this software as just a medium for faster and more efficient typing, a slightly differently built model can also be used to enhance the communication experience of persons with disabilities. Such a model would use the Augmentative and Alternative Communication devices as an API.

In this Python project, we tried to build a text predictor using the concept of n-grams which could be deployed for use where one would want to type something using a hardware or software keyboard. It would process the user input using tri-grams and suggest 3 words which would make the sentence grammatically correct in the order of the decreasing frequency which in turn, is based on the typing habits of the user.

**List of figures**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.no** | **Figure number** | **Figure Name** | **Page number** |
| 1 | 2.1 | Markov chains |  |

**List of tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.no** | **Table number** | **Table Name** | **Page number** |
| 1 | 2.1 | Markov chain probability |  |

**Problem Definition**

One important aspect of effective and fast communication is proper grammar, and choice of words, spellings and more meaningful sentences. In this fast age of electronics and internet, racing our fingers with the speed of our thoughts is error prone.

A text predictor not only focuses on suggesting next words which suit your style, but also to the fact that how much the suggested word would make a complete sense in the context of the information being conveyed through the sentence being typed. In addition to its listed expectations, if a software would be trained to your style of grammar and your diction, and which would adapt the changes of the language you try to adapt in yourself would be an extremely powerful tool and would cater to a wide range of users. Such a tool if provided with the communication service itself would be an added benefit. Also, a text predictor can be an extremely powerful tool when it comes to communication of people with disability. Such a system, allows them to communicate without much effort, and seamlessly and it enhances their current ability to communicate.

A n- gram word predictor solves this problem of less efficient communication by suggesting grammatically sound, and contextually relevant words which are predicted on the basis on the words earlier typed but the user. The degree of n here is 3, hence this is a trigrams implementation which is considered as one of the best values for n, as it is computationally less expensive, easier to develop and maintain, and highly efficient in implementation.

**Introduction**

1. WORD PREDICTION

Since we started to document and type things more often than we wrote, a lot of documentation required less time to create. Eventually this generated a need for faster and more error prone documentation. Word prediction has been a part of this change; in fact word prediction was responsible to drive this change.

Word prediction is used on a lot of platforms: right from your usual keyboard to well-developed apps and cloud services like Gmail, Microsoft Word, Grammarly, Google Docs, etc. A peculiar thing about these services is that you get suggestions which are grammatically precise. To achieve this, large platforms use a huge number of business and literature corpora to train their models. These models get better the more you use them, i.e., they get tailored to your personal writing and documenting style. They train themselves slowly according to the user’s diction, grammar style, spellings. This not only allows you to reduce errors, but over time, also maintains your writing technique. Today, text prediction is used on almost all platforms.

1. USING N-GRAM MODEL TO PREDICT TEXT

N-grams is a very effective model used for text prediction. It retains the grammatical sense while suggesting words, and hence, maintains the semantics of a sentence. It does this by comparing pairs of words and processing which words is followed by what string of word(s). This allows n-grams to predict grammatically correct words, if it is trained under

good quality corpora. The more the quantity and quality of the corpora, better the prediction.

The idea behind the n-grams lies in its formal definition is “a contiguous sequence of n items from a given sample of text”. A n-gram with n = 1 is a unigram, which means it just a single word. A bi-gram predicts the second word on the basis of its previous word. So, the degree of n here is 2. A tri-gram model, which we have implemented here, predicts the third word, on the basis of the first and second word. It predicts the third word by scanning the first two words together, as a single entity. This mere fact, of processing the previous bunch of words together gives it an edge and allows it to maintain semantic and grammatical sense.

The answer of how it predicts that lies in the corpus and the concept of Markov chains. Markov chains, named after Andrey Markov, are mathematical systems that hop from one "state" (a situation or set of values) to another.

For example, if you made a small Markov chain model of 3 sentences :

‘I like to…’,

‘I like that’, and

‘I am a…’ ,

as demonstrated below, you would include "I", "like", "am" and "like", “to”, “that”, “a” as states, which together with other behaviors could form a 'state space': a list of all possible states. In addition, on top of the state space, a Markov chain tells you the probabilitiy of hopping, or "transitioning," from one state to any other state---e.g., the chance that a you’ll get a “to” after “like” .

Figure 2.1: Markov chains

|  |  |  |
| --- | --- | --- |
| Current word | Next possible words | Probability |
| I | (like, am) | (0.25, 0.75) |
| like | (to, that) | (0.90, 0.10) |
| am | a | 1 |

Table 2.1: Markov chain probability

**Implementation**

This project consists of 2 modules, one the GUI application part, and other, the core logic part.

GUI APPLICATION:

**Conclusion and Future scope**

In the project we created, our database was updated with new sentences each time we made a choice based on the predictions, or even if we added a new word of our own choice, thus ignoring the predictions. But, the changes reflect in the corpus after we restart the application. Since, we are using file handling, Python doesn’t update the corpus text file until the program is terminated. This can be, of course, eliminated by developing a database using MySQL. This will make the prediction more fast and robust, as the corpus would immediately be reflected with the changes.

Word prediction, thus, can have a major impact when it comes to using a language in its correct sense. It can be deployed to use for teaching motor skills to children who are learning a new language, but allowing them to grasp the core semantics intuitively, without having the need to write. It allows one to use and understand the language, and document anything with a very less error margin.

**References**

ARTICLES AND BLOGS

## [Coursera - Java Programming: Principles of Software Design (Duke University / Chapter 3: N-grams: predictive text)](https://www.coursera.org/programs/a-p-shah-institute-of-technology-on-coursera-05d2i/browse?collectionId=&currentTab=CATALOG&productId=3-NBikXzEeWwQw6izX918Q&productType=s12n&query=java+duke+university&showMiniModal=true)

1. [Geeks for Geeks - word prediction using n-grams](https://www.geeksforgeeks.org/word-prediction-using-concepts-of-n-grams-and-cdf/)
2. [Kdnuggets - word prediction corpora](https://www.kdnuggets.com/2017/11/building-wikipedia-text-corpus-nlp.html)
3. [StackExchange – predicting the best length of n-grams](https://stats.stackexchange.com/questions/155483/estimating-the-best-length-of-n-gram)
4. [Towards data science – good grams: how to find predictive n-grams for your problem](https://towardsdatascience.com/good-grams-how-to-find-predictive-n-grams-for-your-problem-c04a5f320b39)
5. [StackAbuse – Python for NLP development](https://stackabuse.com/python-for-nlp-developing-an-automatic-text-filler-using-n-grams/)
6. [RealPython – Regex tools](https://realpython.com/lessons/regex-query-tool/)
7. [Tutorials Point – Python regular expressions](https://www.tutorialspoint.com/python/python_reg_expressions.htm)
8. [W3 resource - Sorting a dictionary by values](https://www.w3resource.com/python-exercises/dictionary/python-data-type-dictionary-exercise-1.php?passed=passed)
9. [Medium – Text generation using n-gram model](https://towardsdatascience.com/text-generation-using-n-gram-model-8d12d9802aa0)
10. [Wikipedia - ngrams](https://en.wikipedia.org/wiki/N-gram)
11. [Wikipedia – Markov chains](https://en.wikipedia.org/wiki/Markov_chain)
12. [Brilliant Math & Science Wiki– Markov Chains](https://brilliant.org/wiki/markov-chains/)
13. [Setosa.io - Markov chains: Explained Visually!](https://setosa.io/ev/markov-chains/)
14. [Towards data science- ngrams](https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9)

YOU TUBE

1. [Markov Chains – next word prediction](https://www.youtube.com/watch?v=Ls38CLOvazc)
2. [N-grams in NLP](https://www.youtube.com/watch?v=E_mN90TYnlg)
3. [What are unigrams, bigrams and n- grams – N-gram analysis for ML projects](https://www.youtube.com/watch?v=MZIm_5NN3MY)
4. [N gram model](https://www.youtube.com/watch?v=5263Xqr1YCc&t=464s)