University of Arkansas – CSCE Department
Capstone I – Preliminary Report – Fall 2019

Predictive Typing

Ethan Passmore, Lane Phillips, Layne Bernardo, Roya Rashidi, Sarah Paracha

Abstract

This project is for the company Sorcero. Sorcero lacks phrase completion for users wishing to review and search their corpus documes. Our group sets out to create a predictive phrase typing solution for users reviewing company documents.

We will begin approaching this problem by using Sorcero terminology to analyze their company language and most commonly used words. We will use comparison algorithms to match user inputs to possible words and phrases. Theses algorithms will calculate scores and use the highest score to determine which phrase should most be suggested.

This will efficiently allow users to use less device keys for input writing and help those who may have difficulty expressing what they are searching for in a concise way.

1.0 Problem

The primary use of technological innovation has been to streamline processes to increase efficiency. With the use of keyboard typing to accomplish a number of tasks, such as writing emails or minutes for a meeting, the problem of increasing efficiency exists. In addition, with the age of big data it may become hard to search for relevant results instantaneously for large enterprises and corporations. To solve this problem, predictive typing has been introduced in many applications such as Gmail, iPhone Messaging etc.

Predictive Typing allows individuals to reduce their finger keyboard interaction as the software makes smart predictions as to what the user wishes to type. Such a technology is also important as it increases the reach of users by increasing accessibility. Not having a solution may introduce lag time in typing which can in turn reduce efficiency alongside making technology inaccessible to a group of people.

The area of interest for our project includes predictive typing. However, our aim is to expand the functionality of predictive typing to encompass phrase completion given a corpus of documents. The idea for this functionality is driven by Sorcero, a start-up based in Washington DC.

2.0 Objective

The objective of this project is to develop an autocomplete program for Sorcero. The program is to be designed as to complete the typing of a user who is searching for information within the corpus of documents that Sorcero maintains. The program must complete phrases as the user types, and these phrases must be as accurate and relevant as possible. The intent of Sorcero in asking this is to extend this functionality from only the limited FAQ list, to the entirety of their documents. Once the phrases are recognized and completed they must then take the user to the referenced phrase within the documents. This is meant to make user interaction with regards to searching of Sorcero data as easy, convenient, and user-friendly as possible.

3.0 Background

3.1 Key Concepts

An overarching key idea essential to the completion of this project is an understanding of the field of Natural Language Processing (NLP). NLP is an umbrella term for the amalgamation of computer science, machine learning and linguistics in the processing of human language data. Applications of NLP are all around us in today's time whether that is in our phones through Siri or smart home devices such as Alexa.

This project focuses on the idea of predictive typing, a technology that is mainly used in search engines. It can also be found in text apps such as text messaging or email. With predictive typing, the user starts typing a word or phrase and the app being used auto-completes the word or phrase with example endings that have been used before. Not only does the predictive typing guess what the user is trying to input, it tries to show the most relevant completion text.

The technology involved in producing these results generally consists of a "dictionary" backend, which is used as a base to generate possible phrases, as well as one or more algorithms which produce the actual prediction based on the dictionary. One method of implementing predictive typing involves using a natural language model to weight the possible completions with probabilities based on their likelihood of being used, whereas another (older) tactic is to generate adjacency matrices which can be used to calculate word "proximity," or frequency of use compared to similar input in the past. Other strategies also exist, including the approach of simply recording every phrase when it is searched for and suggesting phrases based on frequency of use as well as the use of machine learning algorithms to generate neural networks capable of making predictions.

All of the methods listed above rely on comparing weights or probabilities of certain words and phrases to select the phrase that is most likely to come next, with the main difference being how the weights of the candidate phrases are generated.

3.2 Related Work

Other developers, notably Google [1], have developed robust and successful predictive typing algorithms used in many applications. Traditionally this relied on the use of adjacency matrices, however it is thought that Google has since switched to a machine learning algorithm to produce text predictions. Libraries also exist which implement predictive typing, including The Embeddable Predictive Text Library Open Source Project [2] and others such as Presage [3] which uses a natural language model and combines the output of multiple algorithms to produce its results. These systems will not be adequate for the project, as Sorcero requires a drop-in, customized predictive typing system tailored for their specific data set and use cases.

4.0 Design

4.1 Design Goals

Until further clarification from Sorcero, the following design will be based on the assumption that a phrase is equivalent to a sentence.

Requirements for the project include:

1) Given a corpus of documents, the software should be able to read all the inputs.
2) Each document read should be split into usable phrases. These phrases will be stored in a constructed database. The database idea may be replaced if a more efficient alternative is identified.
3) The program written should be able to store the frequency of phrases in the document and the expected frequency that a given phrase will be searched.
4) The program may also require implementation of “stemming”.
5) The software should include an implemented search algorithm.
6) The predictive typing technology implemented should contain a user interface which enables phrase completion.

The use cases will include sample documents provided by Sorcero.

4.2 High Level Architecture

The first step of the design will be processing the sample corpus documents provided by Sorcero. This is a two part step, the documents must be read and split up into separate phrases, and then stored for ease of reference by the predictive search.

An algorithm must be written to search through the database to find any and all phrases within the database that match the current text the user is inputting. Once the first word is finished being typed and a space is detected the algorithm will search for all phrases that begin with the same word. And as the users query is typed the same process must be repeated.

The design needs to perform much like Google's search bar, or any other popular search engine, typically functions. When you begin to type your query into Google, a list begins to populate below your search bar. This list contains your personal recent searches that begin in the same manner as seen below in the image. These phrases are things that you have searched in a recent enough time and that match or contain the current contents of the search bar.

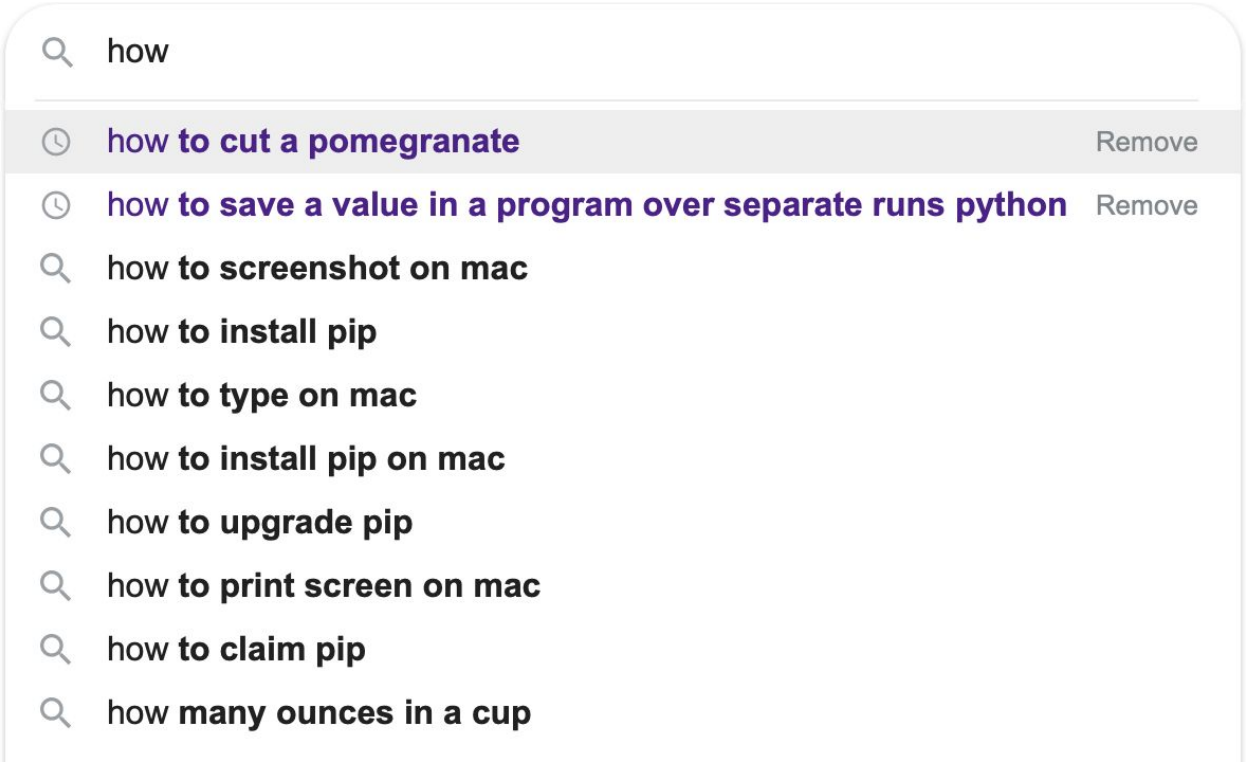

Figure 1: Image to demonstrate Predictive Typing

The program should also have other suggestions based on the same criteria of the contents of the current search bar as it is being typed. These suggestions should be based off of the relevancy of the suggestion. The frequency the suggestion appears within the document or documents being searched and the frequency in a past period that these phrases have been searched for by any other users.

Such more important or more frequently requested phrases should be placed higher in the database of phrases and be given a priority of search by a separate heuristic to help speed up the search for phrases to auto-complete the user's query with better accuracy and speed towards what they are likely to be looking for. Every user search will result in an update in the database to keep up with the most commonly used phrases.

Using text prediction models, we will be able to analyze the most common relationships between words and phrases and thus be able to efficiently predict the next item in a user's input

text. We will analyze these relationships using n-gram models. N-gram models are a type of probabilistic language models that are useful for large sizes of n and have good space-time tradeoffs. A “unigram” is a n-gram of size 1. A “bigram” is a n-gram of size 2. A “trigram” is a n-gram of size 3 and so on and so on… The following graphic is an example of a “2-gram” model.

“Stemming” is a method/tool that will be used to help group words together that contain the same stem but different endings. For example *type*, *typed*, and *typing* all contain the stem *typ*. This method helps reduce the variability of words in the document and help analyze the frequency of words and phrases. In the end, predictive text will help speed up text input processes.

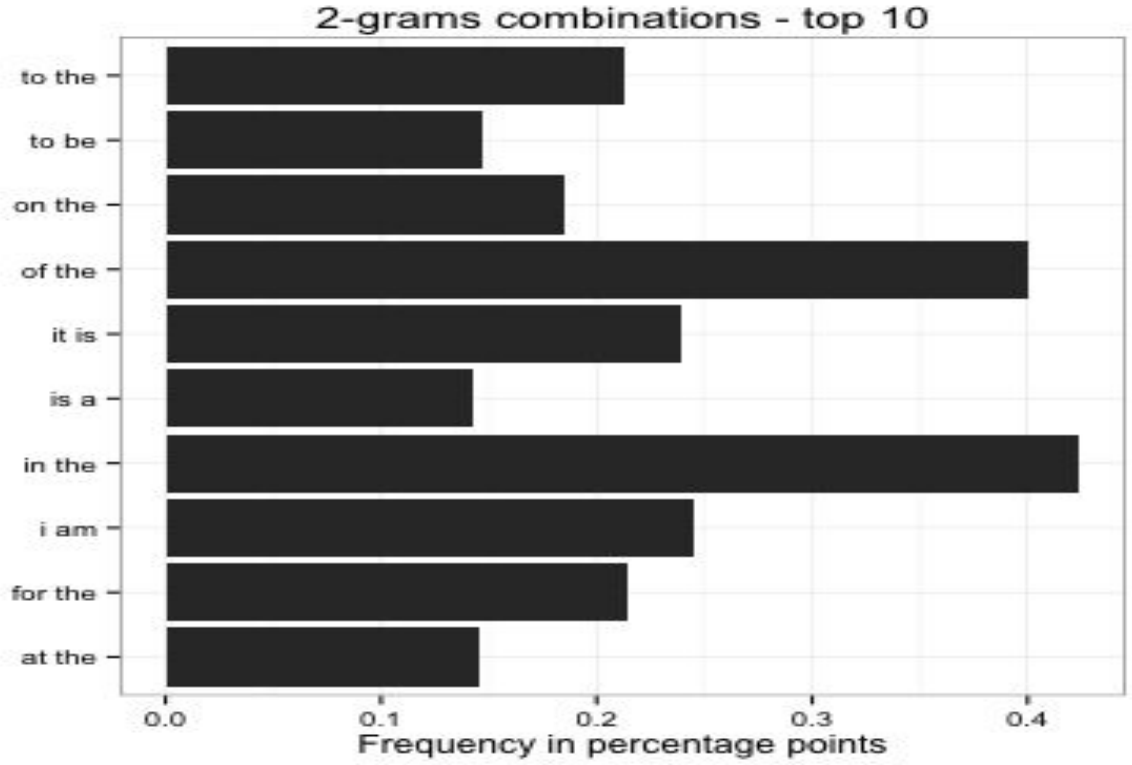

Figure 2: Diagram illustrating 2-gram combinations and their frequency of occurrence

4.3 Risks

Risk	Risk Reduction
Incorrect phrase completion	Ensure phrase completion based on valid metrics such as frequency
Information Loss	Ensure that the program performs no modification on the corpus of documents
Code Injection	Prevent the ability of users to perform code injection attacks via the parser

4.4 Tasks

1. Research existing predictive typing software implementations
2. Clarify project requirements with Sorcero
3. Make final design decisions for our predictive typing technology. This step may include the following design decisions:
    a. Research data storage options. Choose between database or data structures to store phrases
        i. SQL or NOSQL
        ii. In-memory / cache to disk
    b. High- Level understanding of the algorithm to determine the frequency of phrases in the corpus of documents
    c. High-Level understanding of the Search Algorithm
    d. Choose UI Framework
    e. Determine user interaction flow to auto-complete phrase given the correct phrase is identified
4. Implement predictive technology software:
    a. Implement reading of all input files (corpus parser)
    b. Implement storage mechanism for phrases
    c. Implement Frequency Count Algorithm for all phrases
    d. Implement stemming
    e. Implement Search Algorithm
    f. Implement UI
5. Testing (will be performed after implementation of each individual component defined above)
    a. Test corpus parser
    b. Test storage mechanism for phrases
    c. Test Frequency Count Algorithm for all phrases
    d. Test Search Algorithm
    e. Test UI
6. Documentation

4.5 Schedule

Tasks	Dates
1. Research predictive typing approaches 2. Research data storage implementations 3. Clarify specifications with Sorcero	01/20 - 02/03
1. Experiment with test data to choose algorithm 2. Formalize design proposal	02/03 - 02/17
1. Implement corpus parser 2. Implement data storage	02/17 - 03/02
1. Test parser and data storage mechanisms 2. Write unit tests for above 3. Evaluate backend performance	03/02 - 03/16
1. Implement frequency count and stemming algorithms 2. Implement search	03/16 - 03/30
1. Test predictive typing system 2. Test and refine search functionality 3. Write unit tests for above	03/30 - 04/13
1. Create and refine UI 2. Compile documentation 3. Final testing and evaluation	04/13 - 04/27

4.6 Deliverables –

- Design Document:
    - High-level program structure
    - Implementation specifications
    - Usage and configuration documentation
    - UML program diagrams
- Database scheme
    - Either specified in documentation, or integrated into project code
- Project code:
    - Phrase suggestion modules
    - Search module
    - User interface
    - Unit tests
- Final Report
- Research Paper
    - Database schema
    - Algorithm selection

5.0 Key Personnel

Lane Phillips - Phillips is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. Phillips has taken artificial intelligence, information security, computer networks, database systems management, software engineering as that have relevance to the current project. Phillips will be responsible for the database schema and construction. He will also be responsible for a portion of the design and implementation of the phrase search algorithm.

Ethan Passmore – Passmore is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Software Engineering, Programming Paradigms, Computer Networks, Formal Languages and Computability, Database Systems Management, and Algorithms. Passmore will be responsible for implementation of the n-gram models alongside unit testing.

Layne Bernardo - Bernardo is a senior Computer Science major in the Computer Science and Computer Engineering department at the University of Arkansas. He has completed Software Engineering, Operating Systems, Programming Paradigms, and Database Management. He will be responsible for a portion of the phrase search algorithm, documentation, and unit tests.

Roya Rashidi - Rashidi is a senior Computer Science major in the Computer Science and Computer Engineering department at the University of Arkansas - Fayetteville. Rashidi has completed Software Engineering, Networks, Paradigms, and Computer Hardware. She will be responsible for creating a database and implementing algorithm methods such as "stemming".

Sarah Paracha - Paracha is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has taken relevant courses such as Algorithms, Database Management Systems and Software Engineering. She will be responsible for contributing to the corpus parser and the frequency count algorithms alongside unit testing and providing relevant documentation.

Sorcero[4] – Based in Washington DC, Sorcero is a startup company with a focus on natural language processing solutions. Currently, Sorcero collaborates with the Life Science and Insurance industries to enable effective decision making. As a company, Sorcero believes in a vision of leveraging the power of both humans and Artificial Intelligence to empower enterprises.

6.0 Facilities and Equipment

For this project, no facilities and/or purchasable equipment is required. However, we may require a corpus of documents from Sorcero. This corpus of documents will be utilized to test our developed predictive typing technology. Additionally, a corpus of documents may manually be generated by the team or obtained from the internet for testing purposes.

7.0 References

[1] Sullivan, Danny. "How Google Autocomplete Works in Search." Google, 20 Apr. 2018, https://www.blog.google/products/search/how-google-autocomplete-works-search/.

[2] Openhub.net, Massi. "Embeddable Predictive Text Library." Open Hub, Black Duck Software, Inc., https://www.openhub.net/p/lib378.

[3] Presage, https://presage.sourceforge.io/

[4] Sorcero, https://www.sorcero.com/about-us/