University of Arkansas – CSCE Department
Capstone II – Final Report – Spring 2020

Predictive Typing

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

Abstract

Our capstone project is for the company Sorcero. Sorcero is a startup company with a focus on natural language processing solutions[1].

Currently, Sorcero lacks phrase completion for users wishing to review and search their corpus of documents. Our group sets out to create a predictive phrase typing solution that can be expanded to users reviewing company documents.

For the above purpose, our approach includes using n-gram generation and a bucket hashing algorithm. 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[1].

2.0 Objective

The objective of this project is to develop an autocomplete program for Sorcero. The program is to be designed 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. 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 [2], 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 [3] and others such as Presage [4] 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 Requirements and/or Use Cases and/or Design Goals

List of requirements and/or Use cases. This should grow into a comprehensive list which represents all of the requirements that your project must meet which may be described as a list of requirements, a full set of the use cases it will/does satisfy, etc.

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.
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 predictive typing technology implemented should contain a user interface which enables phrase completion.

The use cases to test the program included short speeches such as the Gettysburg Address to Alice in Wonderland, a novel with more than 20,000 words.

4.2 Detailed Architecture

Primary input to the program includes text documents. The first design step is processing the input. This includes parsing the text document and removing any nonessential information or characters. After this, n-grams are generated. N-gram models are a type of probabilistic language model 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.

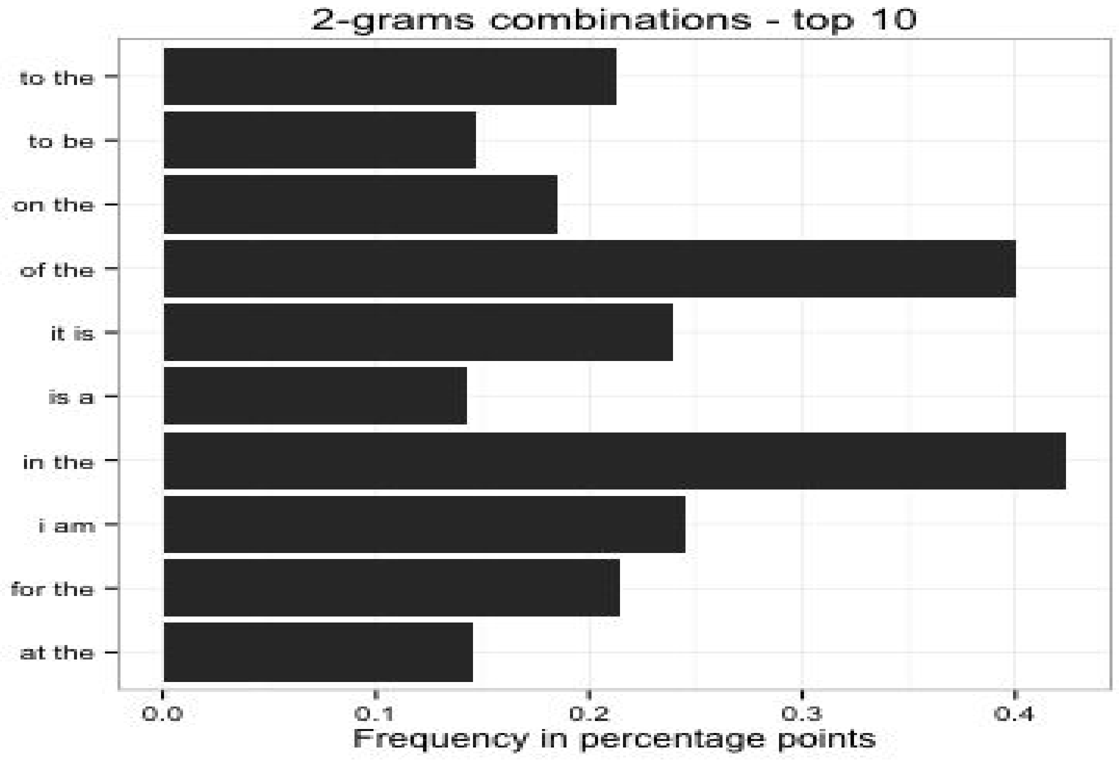


Figure 1: Diagram illustrating 2-gram combinations and their frequency of occurrence[5]

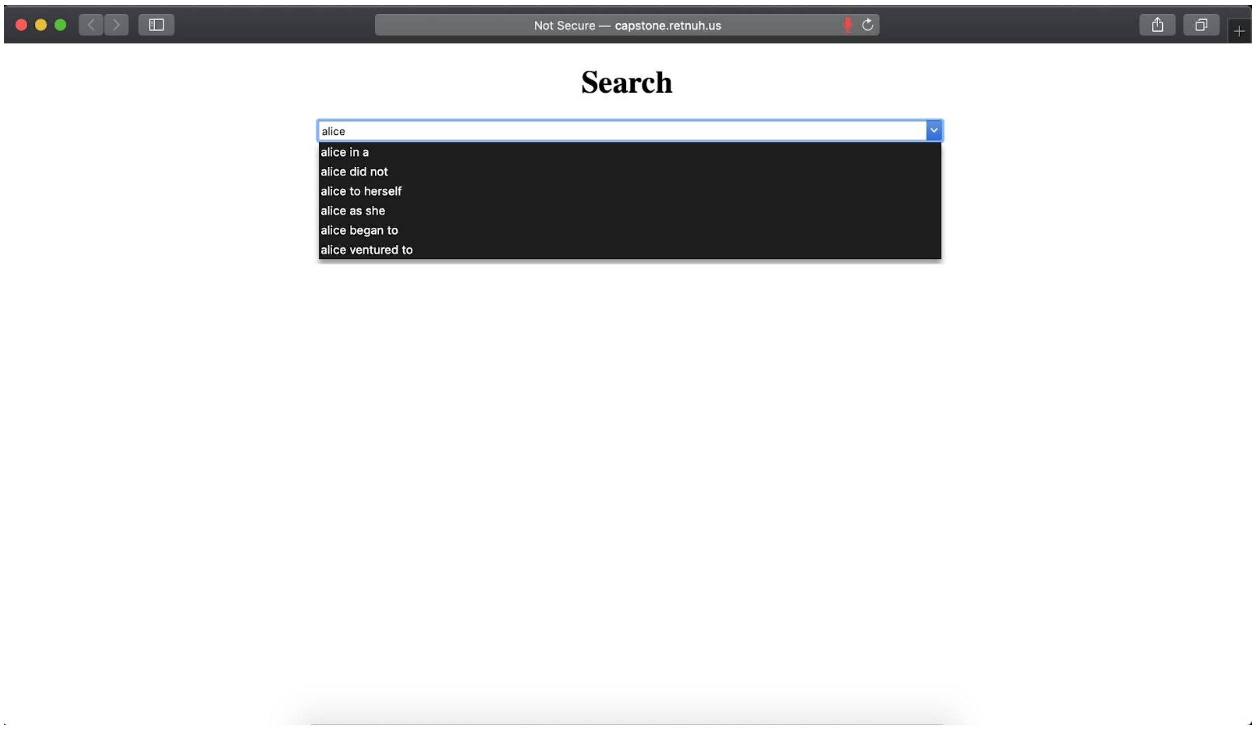
N-grams are used in the program as our text prediction model. Using n-grams, we can analyze the most common occurence of words and thus be able to efficiently predict the next item in a user's input text.

To generate n-grams, the python NLTK (Natural Language Toolkit) library is utilized. This library is used to create a list of all the possible n-grams in the document. Doing so, involves three steps. These steps include firstly tokenizing the parsed string from the text document. Subsequently, a function call is made to generate the n-grams. This function call includes the tokens and the size of the n-grams generated as argument parameters. For our purpose, n-grams of size three i.e. trigrams were initially generated. Later, a modification was made to the function argument to generate 6-grams. Lastly, the n-grams are stored in a list data structure. The n-grams are then pushed to a backend for storage and later used in the phrase prediction process.

For the phrase prediction process, we aimed to write an algorithm that would 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 would search for all phrases that begin with the same word. Continually, as the user's query is typed the same process must be repeated.

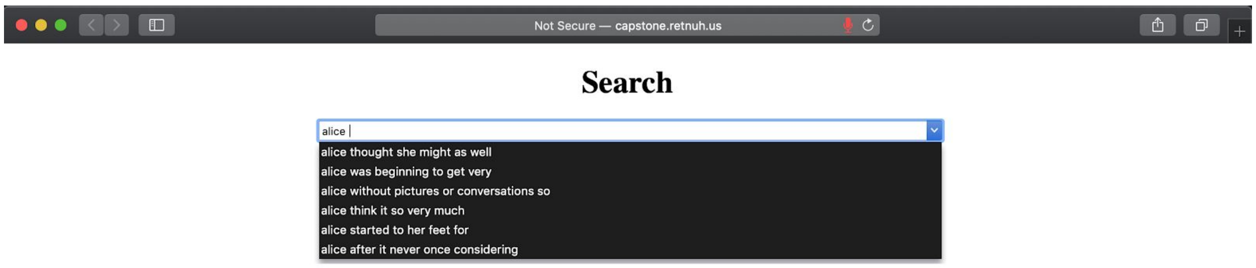
Currently we are using a Python library called Flask to serve the prototype search bar, with a remote MongoDB instance to store the list of n-grams generated in buckets. The database stores the n-grams in buckets categorized by the first word of the generated tri-gram. Every first word of a generated n-gram is considered as the "key" in the hashing process. Within each bucket the n-grams will be sorted by the frequency in which they occur so that when a user begins to type the top of the relevant bucket is what needs to be presented. For example, typing in “Alice” in the search bar will generate “Alice in a” as the top suggestion because it is the tri-gram with the highest frequency in the document that starts with “Alice.”

The front end for our project currently consists of a barebones search bar which takes text input. Upon typing, the search bar then gives the word suggestions to the user based on the given keyword. As illustrated in the below example, the website functions to predict phrases from Alice in Wonderland.



alice
alice in a
alice did not
alice to herself
alice as she
alice began to
alice ventured to

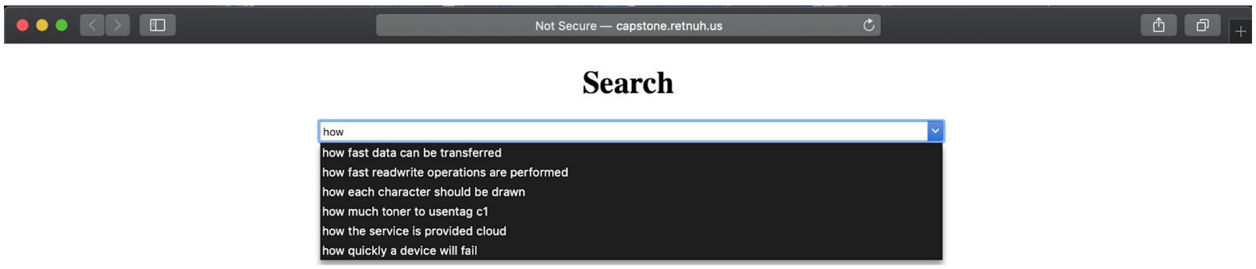
Figure 2: Predictive Typing for the input “Alice in Wonderland” using tri-grams



alice |
alice thought she might as well
alice was beginning to get very
alice without pictures or conversations so
alice think it so very much
alice started to her feet for
alice after it never once considering

Figure 3: Predictive Typing for the input “Alice in Wonderland” using 6-grams

The sample corpus of documents provided by Sorcero was also used as test inputs.

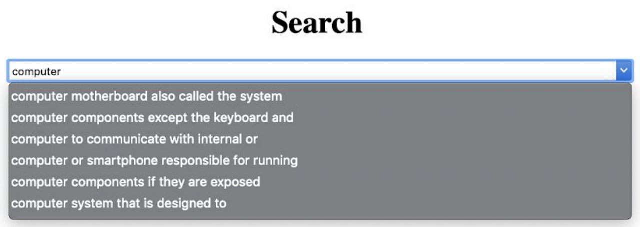

Not Secure — capstone.retnuh.us

Search

how

how fast data can be transferred
how fast readwrite operations are performed
how each character should be drawn
how much toner to usentag c1
how the service is provided cloud
how quickly a device will fail

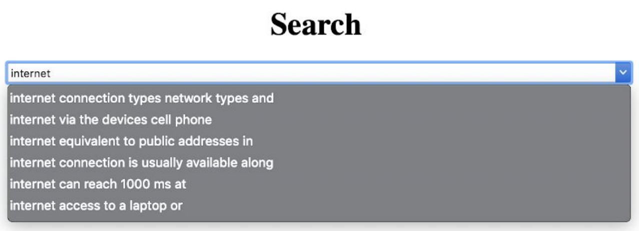
Figure 4: Predictive Typing for the Sorcero's test corpus of documents using 6-grams


Search

computer

computer motherboard also called the system
computer components except the keyboard and
computer to communicate with internal or
computer or smartphone responsible for running
computer components if they are exposed
computer system that is designed to

Figure 5: Predictive Typing for the Sorcero's test corpus of documents using 6-grams



internet connection types network types and
internet via the devices cell phone
internet equivalent to public addresses in
internet connection is usually available along
internet can reach 1000 ms at
internet access to a laptop or

Figure 6: Predictive Typing for the Sorcero's test corpus of documents using 6-grams

The predictive typing program can be improved for a more streamlined user experience and better time-memory tradeoff. For instance, the program can also have other suggestions based on the same criteria of the contents of the current search bar as it is being typed. These suggestions may 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 can all be used as relevancy measures. Additionally, more important or more frequently requested phrases can 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. To do so, every user search may result in an update in the database to keep up with the most commonly used phrases.

4.3 Risks

Risk	**Risk Reduction**
Incorrect phrase completion	The risk was reduced by ensuring that valid metrics such as frequency were the primary determinant in phrase completion. To do so, each bucket sorted the tri-grams in descending frequency.
Information Loss	The risk was reduced by monitoring that the code did not make active changes on the input documents. The documents were simply parsed into strings and only nonessential characters were removed.
Code Injection	This risk was reduced through the implementation of the document parser. The parser currently strips all punctuation, newlines, non ascii text and converts all characters to lowercase. Moreover, the database search is restricted to the

	buckets containing n-grams thus making the attack much more harder.

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 (manual testing performed by running the program)
    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

Note: * tasks not implemented

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 UI 2. Compile documentation 3. Final testing and evaluation	04/13 - 04/27

4.6 Deliverables – Give a thorough listing and description of each item which will be submitted with your final, working project. Each major component should be described. The below is just an example list which should be replaced with your own.

- Final Proposal:
    - High-level program structure
    - Implementation specifications and/or ideas
- Database scheme
    - Either specified in documentation, or integrated into project code
- Project code:
    - Phrase suggestion modules
    - Search module
    - User interface
- Final Report

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 will begin working for Wipro Limited in Plano, Tx in August. 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[1] – 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 was used. However, a corpus of documents from Sorcero was acquired for testing. Additionally, documents were obtained from the internet for testing purposes.

7.0 References

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

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

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

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

[5] RPubs, https://rpubs.com/jcarlosmayo/text_prediction