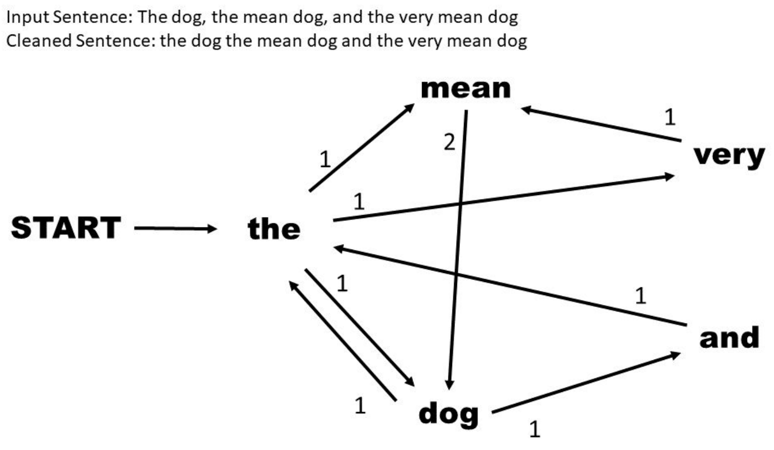
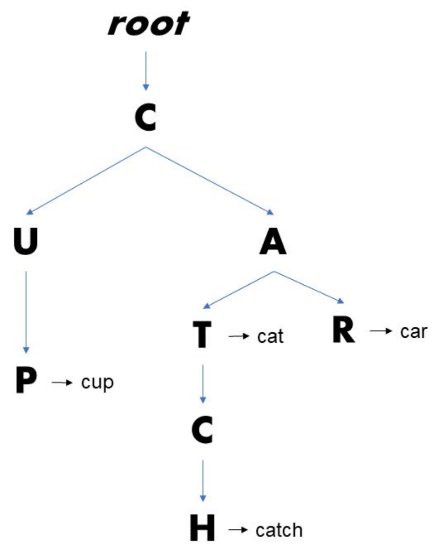
Dorothea French
Spas Angelov
Jake Swartwout

Predictive Text

For this project we mainly used the directed weighted graph data structure in order to store all of our information. Each unique word can be represented as a vertex with directed weighted edges representing how often each word (vertex) follows another. By using the adjacency list implementation, our program has efficient memory usage, as a 2D array implementation would be a sparse matrix. Here is an example of a very simple graph of a simple sentence:


Input Sentence: The dog, the mean dog, and the very mean dog
Cleaned Sentence: the dog the mean dog and the very mean dog
mean
very
START
the
and
dog
1
2
1
1
1
1
1
1

Additionally, the use of a trie improves our search time. The trie is set up as a character tree, creating a path for each word that it encounters. A character node will only be added if it is needed, again saving on space. For instance consider this diagram to reach the word cat:



Our program is intended to function as a word predictor, so we also included saving and loading functionality. In reality, phones keep track of what you type so that they can learn around your writing style. Because we can't keep this application running all the time, loading a save file can quickly load previously gathered data into the graph rather than having to input all the text once again.

Once the data is loaded into the structures, we can make predictions. Our adjacency list is sorted based on the frequency of the word that follows each vertex. This makes it easy to get the most common and least common words. We also included a randomized word which randomly picks one based on the frequency. To make things easier to debug, we also added the investigate word function which displays which words follow an entered word.

Here is a sample output of running our program, feeding in our initial project proposal:

```
Read-o-matic 5000 online.
||1. Read in file                   ||
||2. Open Saved data         ||
||3. Save current data              ||
||4. Display current data           ||
||5. Make random prediction  ||
||6. Predict most common            ||
||7. Predict least common           ||
||8. Investigate word        ||
||9. Quit                           ||
1
Enter the file name you would like me to look at.
proposal.txt
Finished reading in the file
||1. Read in file                   ||
||2. Open Saved data         ||
||3. Save current data              ||
||4. Display current data           ||
||5. Make random prediction  ||
||6. Predict most common            ||
||7. Predict least common           ||
||8. Investigate word        ||
||9. Quit                           ||
5
Using probability to predict the next word
Please enter a word:
the
Hit enter to generate another word.
Press any other key and then enter to exit
the
results
of
writing
ranging…
```

And here are the results in an easier to read paragraph format:

```
The results of writing ranging from project proposals to select
the primary distinction between each with that will be evident
in vocabulary and pair of writing for example growth in order
achieve this implementation is the distinct styles of their own
vertex in the graph will take a trie reduces our...
```

As you can tell, it is not grammatically correct. However, neither are most phone applications. The produced fragment is almost a sentence though, which does mimic keyboard applications. We consider this result successful in meeting our original goal.