Design Document: Autocomplete

Senior Design, Fall 2016

The George Washington University

Joe Paolicelli

We are building an autocomplete system that works to understand the deeper meaning of language input to better understand the user's meaning and provide better predictions. Our system also aims to be faster, more accurate, and more flexible that what is currently available.

Autocompletion of natural language is useful as both a user interface element for a user to directly interact with and as a backend component to improve the speed and accuracy of services. The incorporation of our autocomplete into applications and services will result in improvements in the efficiency and effectiveness in which people use them.

Our system will be able to be incorporated into a wide variety of applications and services, and so indirectly could be used by anyone. Some users will know they are using autocomplete - for example, a user writing a document could be presented with suggestions for upcoming words and phrases using our system. In other cases our system will not be visible to the user, such as when a user uses speech-to-text software and our system is used internally to improve the accuracy of the transcription.

System Components

The steps our system takes to turn raw ngram data into a MongoDB database that provides quick access to applications are shown in Figure 1.

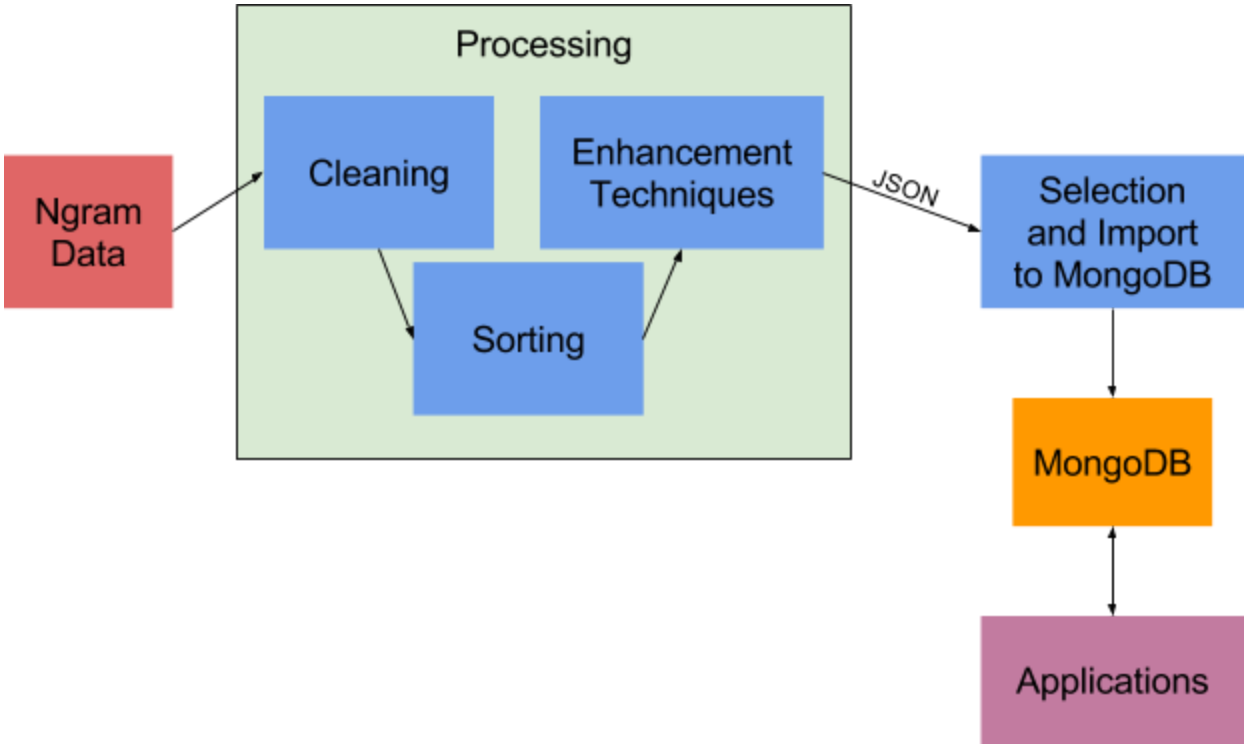

Figure 1

Ngram Data: For our research, we are currently using the ngram dataset offered by Google Books, but any similarly formatted set of ngrams can be used.

Format Example:

example phrase 1978 335 91
example phrase 1979 261 91

(From left to right, the tab separated columns represent the ngram, the year, the total number of appearances, and the number of books appeared in.)

Cleaning: We remove ngrams with tokens that are not wanted (commonly punctuation and other symbols that are not words) and remove unwanted annotations from words (such as part of speech annotations). We also make everything lowercase. Due to the size of the raw ngram datasets, it is necessary to divide the raw files into chunks for processing. Within each chunk, each line is parsed to separate the different fields. The ngram tokens are cleaned, and if the ngram is to be kept, desired fields are saved to a temporary list. At the end of each chunk, the temporary list is sorted alphabetically and saved to a temporary file.

Sorting: Even if the ngram data originally comes sorted (as the Google Books ngram data does), after cleaning the data sorting is likely to become necessary again. All temporary files created from chunks in the previous step are combined and sorted.

Enhancement Techniques: Here we evaluate the ngrams to try and form additional predictions through a variety of methods. The specific methods can vary depending on the dataset being used and the type of predictions desired, but potentially include (but may not be limited to) the following:

- Categorical Classification: Categorize ngrams (particularly those containing proper nouns) and improve predictions based on shared upcoming words or other characteristics of all ngrams classified together.
- Selective augmentation with additional data: Use additional supplementary datasets to add predictions.
- Synonym Substitution: Substitute synonyms to create additional ngrams and predictions.
- Unit conversion: Convert units to create additional ngrams.
- Numeric Reformatting: Convert between digits, numbers in word form, fractions, decimals, and other formats representing numeric amounts to create additional ngrams.
- Part of Speech Analysis: Use the parts of speech of a ngram to improve predictions.
- Location Rescoping: Replace locations with more and/or less specific locations to form additional ngrams.

Processing Export: Predictions for words and phrases are exported as a gzipped JSON file (to allow for easy importing while also saving storage space). Decisions are made as to which word and phrases to export (for example, by limiting the number of predictions for each word to a certain number).

Sample JSON:


```
{
      "word": "vocal",
      "following_frequency": [
            {
                  "text": "in their opposition to",
                  "appearances": 3868
            },
            {
```

```
                    "text": "in their criticism of",
                    "appearances": 2464
               },
               {
                    "text": "and instrumental music and",
                    "appearances": 2362
               }
          ]
     }
```


Selection and Import to MongoDB: The JSON file is opened, and all or some of the data within is imported into a MongoDB database. Within the database, we are currently creating a MongoDB collection for each two letter sequence, and putting the data for all words that start with those two letters into that collection as a MongoDB object formatted similarly to the JSON exported from the processing step. We are examining alternate data structures in case this setup does not give us the desired performance or is deemed undesirable for other reasons.

MongoDB: In MongoDB, indexes and other techniques will be used to ensure the fastest possible performance.

Applications: Applications can query the database by using language specific APIs to MongoDB, querying for the word they desire, and parsing the returned predictions.

Timeline

Basic versions of all components except the Enhancement Techniques component are already complete and working. We are able to make quick predictions on a small subset of data.

During January we will be scaling the amount of data we are using to its full size, and making any changes necessary to keep latency and storage space acceptably low. Changes needed may include data format, MongoDB data structures and indexing, preprocessing changes, and more. Then we will be implementing enhancement techniques to improve result quality, and build a simple demonstration user interface. Specific enhancement techniques and any other further steps have not yet been determined, but will be decided based on the results of previous steps and various testing.