|  |
| --- |
| [Company name] |
| Language Model |
| ICS2203 Natural Language Processing: Methods and Tools Project Task 1 |

|  |
| --- |
| Luke Ellul  [Date] |

Contents

Corpus Processing

The English corpus[[1]](#footnote-1) was used for this language model. The directory **corpus-cleaner** contains **cleaner.js** which includes the code responsible for parsing the xml files of the corpus and deriving the sentences. The derived sentences are saved as plain text files in the directory **cleanTexts** which are then processed by the language modeler.

To derive the appropriate text data from the xml files **cheerio[[2]](#footnote-2)** was used. Cheerio is a Node.js module which is primarily used to scrape content from html pages but can also be used for xml.

Processing Raw Text Files

The file **probFunctions.js** takes care of the mathematics behind the language model. The function **cond** calculates the conditional probability of an event given other events. For example:

**Cond(A, B, C)**

Calculates the probability of A given that B and C also occur. The function accepts two other arguments: **P** and **V**. P is the probability function, the function responsible for measuring the probability of some word occurring in a text file for example. V is a number which is plugged in the formula for Laplace smoothing. In our case, V is the size of our vocabulary.

In our case, we want to calculate the probability of a word occurring given that other words occur before it, or none at all if we want a unigram. This is where **iterateTokens** kicks in. The first argument to **iterateTokens** is **n** which accepts a number that defines the amount of words we want before a word (including the word) to calculate its conditional probability. So, if we have the sentence:

***You are a duck***

and we want to calculate the probability that *duck* occurs given there’s *a* before it we set n to 2. Likewise, we set n to 3 if want to calculate the probability that *duck* occurs given there’s *a* and *are* before it and so on. We set n to 1 if we simply want the probability that there’s *duck*.

The next argument to **iterateTokens** is **tokens** which is simply a list of words. Each of the words will be processed by a probability function along with *n* previous words. The next argument, **P**, is the probability function which is injected into the conditional probability function defined earlier. This function better be fast because it will be called with every single word in the list of tokens.

The next argument is **fn** which is also a function that will be called with the processed data for each token in the list. The processed data includes the conditional probability of a word and its previous words, the word itself, a an array of the *n* words that come before the processed word and the probability of the processed word on its own.

The last argument is **V** which defines the vocabulary size used for Laplace smoothing.

The function **iterateTokens** returns a mapping of the tokens to their processed data. So, for the sentence above, **iterateTokens** will return an array containing the processed data for each word.

1. <http://ota.ox.ac.uk/desc/2554> [↑](#footnote-ref-1)
2. https://www.npmjs.com/package/cheerio [↑](#footnote-ref-2)