**CE706 - short discussion of the solution focussing on functionality implemented and possible improvements and extensions.**

**Parsing**

The parsing conducted using beautifulsoup was done in a generalised way that would work with the two websites provided but also have the capacity to accept the user inputting further websites that could potentially have different html tags/structures. Whilst most of the appropriate text was scraped using the parent name to exclude certain html tags it didn’t work fully which means headers, footers and sidebar information that isn’t relevant made its way into the pipeline. Depending on the time and resource available for scraping the websites and whether you’re aware of which sites need to be scraped this could be refined by inspected the html of the website and navigating using beautifulsoups to reduce excess text/html to improve the indexing.

**Tokenization/POS tagging**

The focus was on sentences when it came to tokenizing as there was a small number of websites being passed through the pipeline. Sentences work better to weight the IDF for the TF-IDF function at the end of the pipeline by treating sentences within the documents as ‘documents’ in the calculation.

Focus on sentences was also due to part of speech (POS) taggers generally not working well when applied to single tokens because they are dependent on the contextual information that is provided by the words surrounding the token. Each sentence was POS tagged separately (rather than all at once) to achieve a more appropriate tag per word/token and then fed into the lemmatizer.

The POS tagger was applied prior to any normalising mainly due to POS taggers performing poorly when sentence structure such as capitalisation/case and punctuation is removed. Even though there is a trade-off POS tagging prior to normalisation helped improve lemmatization.

Improvements could have been made to the mapping of treebank and wordnet to form a more elegant solution. Whilst almost all possible entries have been mapped there could be entries missing which will throw up an error if passed through.

**Lemmatization/Stemming**

There is a trade off with lemmatization and stemming but lemmatization was chosen mainly due to outputting a lexical entry based on understanding of the text. Stemming can suffer from over/understemming where words are chopped back too far/short which can change the meaning of the term e.g. caring into car. Lemmatization requires linguistics support in analysing text which meant that there had to be more effort put into parts of speech tagging as mentioned above and can generally take a lot more time than applying a stemmer.

An improvement that could be made starts at the parsing as ‘noisy’ text made it into the sentence structure which would influence the POS tagging and lemmatization.

**Pre-processing/clean up**

The pre-processing was conducted after the POS tagging and lemmatization mainly to retain the sentence structure to support the POS and lemma in understanding the context the words were being used in. Several processing techniques were used in the program to clean the text as much as possible so that uninformative words/characters didn’t make it into the TF-IDF. Improvements could be made with stop words, ensuring that the package is extensive enough to ensure all possible stop words are being removed.

Extensions to the processing could include different languages which may require packages installed to normalize different characters.