Instructions:

1. **Importing Libraries :** Following are the libraries that I am going to import to take help in Cleaning text data.
    - Nltk - It provides easy-to-use interfaces along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.
    - String - Here I use the help of string for using punctuation lists and convert words to lowercase.
    - Re - It is used to remove the regular expressions which are not useful in text.
    - Autocorrect - I import it to check whether the accessed word is correctly spelled (if not then correct it).
    - Pandas - To access and read dataframe.

2. Here I defined all the functions to clean the text data
    - remove_html_nums() is used to remove html tags from the text like "<head><body>" using regex.
    - Strip_punctuation is defined for removing punctuation from the text like ".?!" and also the symbols like "@#$".
    - lemmatizing() is very similar to stemming. The only difference is that lemmatization tries to do it the proper way. It doesn't just chop things off, it actually transforms words to the actual root.
    - stop_words() - Stop words are a set of commonly used words in a language. In my experience, stop word removal, while effective in search and topic extraction systems, showed to be non-critical in classification systems. However, it does help reduce the number of features in consideration which helps keep your models decently sized.
    - stemming() - Stemming is the process of reducing inflection in words to their root form. The "root" in this case may not be a real root word, but just a canonical form of the original word.
    - spell_check() - helps to correct the incorrect spelled words.
    - Tokenization - Tokenization is the process of turning sensitive data into nonsensitive data called "tokens" that can be used in a database or internal system without bringing it into scope.
    - Lowering() - Lowercasing ALL your text data, although commonly overlooked, is one of the simplest and most effective forms of text preprocessing.

3. Defined Functions to calculate useful Variables from given text data.
4. Importing Master Dictionary and Text Data - Here we access the master dictionary and store it in Master_dict. Taking only important columns (positive, negative,constraining, uncertainty) and store it in dictnry.
5. Here defining a function to run our input files(name as file_namefile_number) in a for loop and storing data in a dataframe named df and returning the dataframe. Due to large text data it approximately takes 5 seconds to run a file while approximately **12 minutes** to run the whole code.