1 Simple Feature Extraction

For the feature extraction, I use a list of token counts. The function iterates over each token and checks whether it exists in the global feature dictionary, if it doesn't, a new entry is created with the next available index. This index is used to reference the feature in the feature vector. The index is then used to update the feature vector for the current document. If the feature doesn't exist, it's initialized with a count of 1, otherwise its count is incremented by 1.

2 Cross-Validation on Training Data

In the cross-validation function, the dataset is divided into 10 folds. In each iteration, one fold is used as the validation set while the remaining folds are used for training. This process repeats until every fold has served as the validation set exactly once. For each iteration, performance metrics such as precision, recall, F1-score, and accuracy are computed and stored. Finally, these metrics are averaged across all folds to obtain an overall estimate of the model's performance.

3 Error Analysis

Using the training data we run an error analysis of the first fold from the cross validation, meaning the first tenth of the training data is for validation while the rest is for validation. For the positive class, there are 212 false negatives and 255 false positives. Naturally as a binary classification the values for negatives are reversed.
Printing out the features sorted by most common from the false positives and false negatives then finally a sum of the counts we get the top error features. The top features in both false positive and false negatives for the positive class include common words such as "the", "to", "I", "a", and "in". This may indicate that the stop words need to be removed as they are interfering with the classification by introducing noise. Further observations tell us that the actually negative class has the lower values across the scores precision (0.726), recall (0.688), and f1 (0.707) as opposed to the actually positive class. Recall is the lowest due to high false negatives which are the actual negatives classified as positives. First observation is that there is a lot of punctuation that interferes with the tokenization as these individual words will be considered as one long token.

- Sir, there are enemies in the White House.Something needs to be done TODAY not November 6th.Clinton,Obama,Panetta,Petraeus,Lamb,Lew,Donilon
- Sunday night wrap-up: Manning, Broncos roll Saints: His arm might not be full strength yet. And then he whacked ... http://t.co/fkWUIfYR

4 Optimising pre-processing and feature extraction

The first and simplest improvement, normalizing by lowercasing all tokens during preprocessing, which resulted in a 0.1 increase in average scores from baseline. Subsequent enhancements focused on lemmatization, stop word removal, and eliminating characters such as punctuation and hashtags further standardize the text. Other enhancements like POS tagging and the removal of links or mentions led to a decline in

performance. Other enhancements were made at the feature level such as bigrams to capture relational information, number of words, and SVM tuning. While POS tagging and trigrams also led to a decline in performance. After some analysis I returned to update the clean up instead keeping exclamations and question marks to keep emphasis and replacing them along with hashtags as tokens to remove noise. Finally experiments with expanding contractions and removing enclosing characters such as brackets and parenthesises also degraded performance. After implementing NLTK word_tokenize and reducing elongated words to standardize the text further, a large jump in performance came from using NLTK's Sentiment Intensity analyzer and its scores as features, followed by a small increase from removing the <emph> token to let the analyzer use the punctuation as emotional signals. The final improvement to a 0.8890 f1 score was to use a logistic regression classifier.

Pre Processing Enhancements	Precision	Recall	F1-Score	Accuracy
Lowercasing	0.8367	0.8382	0.8369	0.8382
WordNetLemmatizer	0.8370	0.8384	0.8372	0.8384
Stop Word Removal	0.8383	0.8398	0.8385	0.8398
Part of Speech tagging preprocessing	0.8342	0.8355	0.8345	0.8355
Link Removal	0.8375	0.8388	0.8376	0.8388
Removed Punctuation ? | ! | : | ; | .	0.8409	0.8419	0.8411	0.8419
Removed link and @mentions	0.8425	0.8432	0.8427	0.8432
Part of Speech tagging feature	0.8449	0.8454	0.8450	0.8454
Bigrams feature	0.8653	0.8667	0.8651	0.8667
Trigrams feature	0.8648	0.8661	0.8637	0.8661
Number of words feature	0.8654	0.8668	0.8653	0.8668
SVM Param Balanced Weight	0.8656	0.8670	0.8656	0.8670
Opinion Lexicon Count feature	0.8744	0.8754	0.8745	0.8754
Replaced ! | ? with <emph> and hashtags with <hashtag> tokens	0.8772	0.8782	0.8773	0.8782
Expanded contractions preprocessing	0.8760	0.8770	0.8761	0.8770
NLTK Tokenizer preprocessing	0.8789	0.8798	0.8791	0.8798
Reduce elongated words preprocessing	0.8794	0.8802	0.8796	0.8802
Sentiment Intensity Analyzer feature	0.8862	0.8868	0.8863	0.8868
Removed <emph> and ; preprocessing	0.8873	0.8880	0.8874	0.8880
Logistic Regression Classifier	0.8892	0.8890	0.8890	0.8890