COMP 550 Assignment 1

Daniel Korsunsky, ID: 260836959

October 1, 2021

I. The problem setup

The experimental task was to train an ML model to distinguish and classify sentence-long movie reviews into either a positive or negative sentiment. The data[1] used was categorized by positive or negative polarity, with 5,331 sentences each. The research question explored in this experiment was: *What preprocessing decisions work well for sentence-level sentiment classification?*

II. The experimental procedure

The positive and negative sentiments were loaded and concatenated into a single array. A parallel second array of category labels was created, with 1s corresponding to positive reviews and 0s corresponding to negative reviews. The movie review data and corresponding labels were then shuffled (with a random seed of 550) and split 80/20 into train and test data, respectively.

The text classifier pipeline was constructed using sklearn's `CountVectorizer` feature extraction class which converted the text to a matrix of token counts, which was then passed to the `LogisticRegression` classifier. This pipeline, along with the pre-processing parameters (see Section III for parameter details), was subsequently fed into sklearn's `GridSearchCV` classifier, which used 5-fold cross-validation on the training set to evaluate model accuracy across each possible parameter combination. The highest-scoring model was then reported and tested on the test set. Finally, sklearn-metrics's classification report was used to report the accuracy and precision, recall, and f1-score on both the positive and negative reviews.

III. Parameter settings

All preprocessing decisions were implemented within the pipeline as parameters of `CountVectorizer`, which converted the text to a matrix of token counts:

- `tokenizer`: the option to use the default tokenizer or the LemmaTokenizer, which was manually implemented as a class using NLTK's `word_tokenize` function which omitted NLTK's English stopwords corpus
- `token_pattern`: the option between the default setting (which only accounted for tokens of 2+ alphanumeric characters and removed punctuation) and one which included all punctuation in the raw data
- `stop_words`: the option to omit NLTK's stopwords corpus from the data
- `ngram_range`: the option among unigrams, bigrams, or a range of 1-2-grams

IV. Results and conclusions

The best-performing model achieved an **CV score of 0.766 and test accuracy of 0.7595** (and the results seen in the metrics to the right) with the following parameters:

	precision	recall	f1-score
positive	0.75	0.78	0.76
negative	0.77	0.74	0.75

`ngram_range:` **`(1, 2)`**`, stop_words:` **`None`**`, token_pattern:` **`default`**`, tokenizer:` **`None`**

[1] Bo Pang and Lillian Lee, Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, *Proceedings of ACL 2005.*

While the (1, 2)-gram range improved model performance on the validation set, it did not improve test accuracy. The table below shows the test accuracy of introducing one non-default preprocessing decision to the default pipeline. All non-default parameters decreased model performance[2], with the bigram parameter significantly decreasing performance. *Thus, the best preprocessing decisions for sentence-level sentiment classification were (1, 2)-grams, excluding punctuation, and not accounting for stopwords or lemmatizing words.*

Parameter	default	bigram	(1, 2)-gram	punctuation incl.	stopwords	lemmatization
Accuracy	0.7599	0.7074	0.7595	0.7590	0.7468	0.7450

There are many possible explanations for the decreases in test accuracy. For stopwords, words like "above" and "below" could have correlated with a reviewer's expectations, and thus their omission would have obfuscated sentiment. In lemmatization, information like word tense is lost, which could also have decreased sentimentality. The larger decrease in the bigram model accuracy was more surprising because bigrams could theoretically carry more information and better account for context. It is possible that the increased number of features could have led to overfitting.

V. Limitations of the study

Because the models relied solely on word frequency features for their predictions, the context of the words—and other information besides word frequency—was largely lost. The order of words (which could give different meanings to the sentence) was lost as well: for example, "I don't love this film. I actually hate it" and "I don't hate this film. I actually I love it" would have the exact same feature vectors. Other levels of linguistic expression, such as sarcasm, were also well beyond the model's comprehension. The restriction to the LogisticRegression model and word frequency meant that these features of language (e.g. word order, sarcasm) would be extremely difficult to capture.

[2] It must be mentioned that although the `LemmaTokenizer` excluded stopwords, when both lemmatization and stopwords were used as model parameters, several stopwords were still lemmatized at runtime and thus never removed. While I could not resolve this issue, the fact that both preprocessing decisions decreased model performance suggests that it would be unlikely that a pairing of the two would have increased model performance.