Kaye Ng
Vivian Uy
INTRNLP S17

Programming Exercise #4

Corpus (Dictionary) Files

1. Slang.txt - contains the expanded forms of commonly-used slangs
2. Contractions.txt - contains the expanded forms of contracted words in English
3. Virgin America and US Airways Tweets.csv - contains the airline, sentiment and text from 3,462 tweets, of which 2,444 are negative sentiments, 552 are neutral and 421 are positive sentiments.
4. Frequency_dictionary_en_82_765.txt - contains a list of English words along with their frequency counts

Methodology

Text Preprocessing

1. Removed hashtags
    - Hard to extract info if it contains multiple words (ex. '#NotHappyToday')
2. Removed hyperlinks
    - Don't contain relevant information for sentiment analysis
3. Excluded emoticons
    - Hard to interpret sentiment from emoticons
4. Reduced any repeated instances of letters to two repetitions at most (ex. 'cooooool' to 'cool')
    - To make it closer to its original form for spelling checker to have higher chance of correcting it to the right word
5. Removed punctuation marks
    - Don't have any bearing in determining the sentiment of the text
6. Expanded contracted words and slang words if found in the dictionary
    - To see if using expanded forms would increase the classifier's performance
7. Checked for the correct spelling of each token
    - To ensure that tokens with the same spelling will not be missed
    - To limit the number of features
    - To ensure the correctness of the features
8. Lemmatization
    - To see if using the base form of words would increase the classifier's performance

The effects of preprocessing on the model depends on which techniques were used, but it's expected that it would boost the accuracy because less informative information is filtered out. However, when we compared the results of preprocessing the text and leaving it raw, taking the raw text as features actually led to a significant difference in the performance of the classifier, as shown in Tables 1 to 3. This might be due to a potential loss of information that resulted from excluding certain features, which led to a decrease in the classifier's performance.

Table 1. Comparison of weighted average F1 scores for processed and unprocessed text, with no resampling.

	Feature Extraction			
	Bag of words	Binary bag of words	TFIDF	Binary TFIDF
Preprocessing	**0.633446**	0.629974	0.589608	0.589608
No preprocessing	**0.720118**	0.716752	0.587191	0.582473

Table 2. Comparison of weighted average F1 scores for processed and unprocessed text, with random undersampling.

	Feature Extraction			
	Bag of words	Binary bag of words	TFIDF	Binary TFIDF
Preprocessing	0.705256	**0.707579**	0.675712	0.663360
No preprocessing	0.764993	0.763209	**0.770091**	0.762715

Table 3. Comparison of weighted average F1 scores for processed and unprocessed text, with near-miss undersampling.

	Feature Extraction			
	Bag of words	Binary bag of words	TFIDF	Binary TFIDF
Preprocessing	0.391377	0.414983	0.660755	**0.663961**
No preprocessing	0.553062	0.584741	**0.762890**	0.757245

Train-Test Split

The data was split into a 70% train set and a 30% test set. The results were tested with no resampling, random undersampling and near-miss undersampling methods.

Feature Extraction

After testing multiple times with different parameters, we settled on using a range of unigrams to trigrams and a document frequency threshold of 90% since they give the best results among all the other configurations we tested, namely using a 50% threshold, including a minimum document frequency filter of 5%, and trying out ngram ranges of unigrams to bigrams and bigrams to trigrams.

Unexpectedly, implementing random undersampling proved to have the best results out of the three test cases, since it's the most naive way of selecting samples by the target class. We also found out that using certain combinations of feature extraction methods and sampling methods affects the classifier's performance. For example, the bag of words model works better with random undersampling than near-miss undersampling (which works better with TF-IDF).

The results of the model when using the binary representation of bag of words and TF-IDF are quite similar to the results from using bag of words and TF-IDF by themselves.

Words that aren't verbs, adjectives or adverbs were removed as features because these POS tags convey emotions.

Model

Multinomial Naive Bayes is a generative algorithm based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of a feature. Bayes theorem calculates the probability $p(y|x)$ where y is the class of the possible outcomes and x is the given instance which has to be classified, representing certain features.

Limitations

- Hashtags containing potentially useful information are ignored.
- The dictionaries used do not provide a comprehensive list of slangs and contractions.
- Emoticons that contain valuable information regarding sentiment were left out in the process of analysis.
- The size of the corpus is not sufficient enough for the classifier to learn from.
- The context is not being considered in the training, since tweets are of short-form.
- Sarcasm, irony, humor, and other text with implied meaning which affect the tone of a text can't be detected.
- False negatives, where the model sees a negative word like "crap" but doesn't classify it as positive based from the overall context (ex. "Holy crap! I loved this!")
- Cultural differences also affect the sentiment of a text, such as by one's choice of words.

References

Bird, S., Loper, E., & Klein, E. (2009). *Natural language processing with Python.* California, USA: O'Reilly Media Inc.

List of English contractions. (2019). In *Wikipedia*. Retrieved March 18, 2018, from https://en.wikipedia.org/w/index.php?title=Wikipedia:List_of_English_contractions&oldid=888251907.

mammothb. (2018). symspellpy [GitHub repository]. Retrieved from https://github.com/mammothb/symspellpy

Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J. Bethard, S.J., & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60. Retrieved March 17, 2019, from http://nlp.stanford.edu/pubs/StanfordCoreNlp2014.pdf.

B. Omondi Ochieng, S., Loki, M., & Sambuli, N. (2016). Limitations of Sentiment Analysis on Facebook Data. *International Journal of Social Sciences and Information Technology* 2(4), 425-433.

Princeton University. (2010). WordNet. Retrieved March 17, 2019, from https://wordnet.princeton.edu.

Verma, R. (2018). SMS/Message Slang Translator [GitHub repository]. Retrieved from https://github.com/rishabhverma17/sms_slang_translator.

Appendix A. Some test runs with classification scores

Bag of words with random undersampling

>> max_df=0.5

1. ngram_range=(1, 3)

	precision	recall	f1-score	support
negative	0.816156	0.812760	0.814454	721
neutral	0.357576	0.339080	0.348083	174
positive	0.524476	0.572519	0.547445	131
micro avg	0.701754	0.701754	0.701754	1026
macro avg	0.566069	0.574787	0.569994	1026
weighted avg	0.701143	0.701754	0.701270	1026

2. ngram_range=(1, 2)

	precision	recall	f1-score	support
negative	0.835052	0.786408	0.810000	721
neutral	0.400000	0.367816	0.383234	174
positive	0.443850	0.633588	0.522013	131
micro avg	0.695906	0.695906	0.695906	1026
macro avg	0.559634	0.595937	0.571749	1026
weighted avg	0.711322	0.695906	0.700854	1026

3. ngram_range=(2, 3)

	precision	recall	f1-score	support
negative	0.818182	0.748960	0.782042	721
neutral	0.382979	0.413793	0.397790	174
positive	0.443820	0.603053	0.511327	131
micro avg	0.673489	0.673489	0.673489	1026
macro avg	0.548327	0.588602	0.563720	1026
weighted avg	0.696577	0.673489	0.682311	1026

Bag of words with random undersampling
>> max_df=0.9

4. ngram_range=(1, 3)

	precision	recall	f1-score	support
negative	0.863850	0.765603	0.811765	721
neutral	0.397129	0.477011	0.433420	174
positive	0.466292	0.633588	0.537217	131
micro avg	0.699805	0.699805	0.699805	1026
macro avg	0.575757	0.625401	0.594134	1026
weighted avg	0.733938	0.699805	0.712547	1026

Bag of words with near-miss undersampling

5. max_df=0.9, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.882353	0.332871	0.483384	721
neutral	0.285311	0.580460	0.382576	174
positive	0.240000	0.732824	0.361582	131
micro avg	0.425926	0.425926	0.425926	1026
macro avg	0.469221	0.548718	0.409180	1026
weighted avg	0.699084	0.425926	0.450736	1026

6. max_df=0.5, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.812057	0.317614	0.456630	721
neutral	0.241042	0.425287	0.307692	174
positive	0.233410	0.778626	0.359155	131
micro avg	0.394737	0.394737	0.394737	1026
macro avg	0.428836	0.507176	0.374492	1026
weighted avg	0.641336	0.394737	0.418926	1026

Tf-idf with near-miss undersampling

7. max_df=0.5, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.876344	0.678225	0.764660	721
neutral	0.341991	0.454023	0.390123	174
positive	0.417722	0.755725	0.538043	131
micro avg	0.650097	0.650097	0.650097	1026
macro avg	0.545352	0.629324	0.564276	1026
weighted avg	0.727166	0.650097	0.672208	1026

8. max_df=0.9, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.868687	0.715673	0.784791	721
neutral	0.345382	0.494253	0.406619	174
positive	0.469945	0.656489	0.547771	131
micro avg	0.670565	0.670565	0.670565	1026
macro avg	0.561338	0.622138	0.579727	1026
weighted avg	0.729028	0.670565	0.690394	1026

Bag of words with no preprocessing
>> no undersampling

9. max_df=0.9, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.746098	0.994452	0.852556	721
neutral	0.774194	0.137931	0.234146	174
positive	0.941176	0.244275	0.387879	131
micro avg	0.753411	0.753411	0.753411	1026
macro avg	0.820489	0.458886	0.491527	1026
weighted avg	0.775770	0.753411	0.688350	1026

10. max_df=0.5, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.712451	1.000000	0.832083	721
neutral	0.333333	0.005747	0.011299	174
positive	1.000000	0.083969	0.154930	131
micro avg	0.714425	0.714425	0.714425	1026
macro avg	0.681928	0.363239	0.332771	1026
weighted avg	0.684870	0.714425	0.606427	1026

Bag of words with no preprocessing
>> with random undersampling

11. max_df=0.9, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.878655	0.833564	0.855516	721
neutral	0.500000	0.500000	0.500000	174
positive	0.571429	0.732824	0.642140	131
micro avg	0.764133	0.764133	0.764133	1026
macro avg	0.650028	0.688796	0.665885	1026
weighted avg	0.775212	0.764133	0.767980	1026

12. max_df=0.5, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.872868	0.780860	0.824305	721
neutral	0.442105	0.482759	0.461538	174
positive	0.502618	0.732824	0.596273	131
micro avg	0.724172	0.724172	0.724172	1026
macro avg	0.605864	0.665481	0.627372	1026
weighted avg	0.752541	0.724172	0.733668	1026

Tf-idf with no preprocessing
>> no undersampling

13. max_df=0.9, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.708948	1.000000	0.829689	721
neutral	0.333333	0.005747	0.011299	174
positive	1.000000	0.045802	0.087591	131
micro avg	0.709552	0.709552	0.709552	1026
macro avg	0.680760	0.350516	0.309527	1026
weighted avg	0.682409	0.709552	0.596147	1026

14. max_df=0.5, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.712451	1.000000	0.832083	721
neutral	0.333333	0.005747	0.011299	174
positive	1.000000	0.083969	0.154930	131
micro avg	0.714425	0.714425	0.714425	1026
macro avg	0.681928	0.363239	0.332771	1026
weighted avg	0.684870	0.714425	0.606427	1026

Tf-idf with no preprocessing
>> with random undersampling

15. max_df=0.9, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.900813	0.768377	0.829341	721
neutral	0.450495	0.522989	0.484043	174
positive	0.516746	0.824427	0.635294	131
micro avg	0.733918	0.733918	0.733918	1026
macro avg	0.622685	0.705264	0.649559	1026
weighted avg	0.775406	0.733918	0.746006	1026

16. max_df=0.5, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.893864	0.747573	0.814199	721
neutral	0.428571	0.517241	0.468750	174
positive	0.478873	0.778626	0.593023	131
micro avg	0.712476	0.712476	0.712476	1026
macro avg	0.600436	0.681147	0.625324	1026
weighted avg	0.761969	0.712476	0.727375	1026

Bag of words in binary representation with random undersampling

17. max_df=0.9, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.839580	0.776699	0.806916	721
neutral	0.368932	0.436782	0.400000	174
positive	0.529412	0.618321	0.570423	131
micro avg	0.698830	0.698830	0.698830	1026
macro avg	0.579308	0.610600	0.592446	1026
weighted avg	0.720160	0.698830	0.707712	1026

18. max_df=0.5, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.822254	0.789182	0.805379	721
neutral	0.356688	0.321839	0.338369	174
positive	0.435028	0.587786	0.500000	131
micro avg	0.684211	0.684211	0.684211	1026
macro avg	0.537990	0.566269	0.547916	1026
weighted avg	0.693857	0.684211	0.687187	1026

Tf-idf in binary representation with random undersampling

19. max_df=0.9, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.860465	0.718447	0.783069	721
neutral	0.355102	0.500000	0.415274	174
positive	0.469274	0.641221	0.541935	131
micro avg	0.671540	0.671540	0.671540	1026
macro avg	0.561614	0.619889	0.580093	1026
weighted avg	0.724813	0.671540	0.689906	1026

20. max_df=0.5, ngram_range=(1,3)

	precision	recall	f1-score	support
negative	0.834633	0.742025	0.785609	721
neutral	0.357488	0.425287	0.388451	174
positive	0.466292	0.633588	0.537217	131
micro avg	0.674464	0.674464	0.674464	1026
macro avg	0.552804	0.600300	0.570426	1026
weighted avg	0.706684	0.674464	0.686540	1026