COM3110 Text Processing

Sentiment Analysis Report

Lea Button

A report detailing the implementation of my final solution and describing the decisions made for feature selection. Results are discussed, and the best models are evaluated.

December 22, 2023

1 Implementation details

1.1 Classifier

A Multinomial Naive Bayes Classifier with Laplace Smoothing has been implemented in *classifier.py*, using the below decision:

$$s^* = \underset{s_i}{argmax}\ p(s_i)\prod_{j=1}^{N} p(t_j|s_i)$$

$$p(t_j|s_i) = \frac{count(t_j, s_i) + \alpha}{(\sum_f count(t_f, s_i)) + \alpha|V|}$$

Where $\alpha > 0$ is the Laplace smoothing parameter, typically 1.

The calculation for prior probabilities can be found in the *train* function (line 29). The calculation for the likelihoods of each feature for each class can be found in the *calculate_likelihoods* function (line 91). The calculation of the posterior probabilities of each class for a given phrase can be found in the *calculate_probabilities* function (line 62).

A Utils class has been implemented in *utils.py*, containing functions to process the datasets, apply various preprocessing techniques, calculate macro-f1 score, save predictions to files, and generate heatmaps. The calculation for macro-f1 score can be found in *calculate_macro_f1* function (line 227).

1.2 Preprocessing

The following preprocessing methods have had full functionality implemented:

- Lowercasing (LC)
- Stoplisting (SL) *where the stoplist used was adapted from the NLTK [1] english stopwords list to suit the application more.*
- Expanding abbreviations (ABR) *where abbreviated terms are replaced with their full versions.*
- Negation (NEG) *where 'not_' is appended to all words following a set list of negation terms.*
- Binarisation (BIN)
- Stemming (STEM) *where word affixes are removed.*
- Lemmatization (LEM) *where words are reduced similar to stemming, but all reductions result in real language words.*

While their functionalities are there, not all methods will be used in the final *best models* - through empirical testing, optimal combination(s) of the preprocessing methods where found (discussed in more detail further into the report).

1.3 Baseline results

Results for 3-class and 5-class for both all_words and features are 0.485913 and 0.284448 respectively which is already an improvement over the majority class baseline. These results serve as a baseline to determine the improvements (or deteriorations) that different feature selection methods, and preprocessing combinations have on the model(s).

2 Feature Selection

The final feature selection method used is Chi^2, which was determined through the investigation of 3 initial methods, one of which was Chi^2. The remaining two where TF-IDF with thresholds, and selecting nouns, verbs, adjectives and adverbs (NVAR).

TF-IDF was selected as an initial method because of its ability to capture the relevance of a term in a phrase. However, TF-IDF does not consider terms in relation to each other, or in relation to sentiment. Calculation of TF-IDF values was implemented by scratch, to allow for easy investigation into changing the TF moderation method. However, experimental results showed very minimal improvement when using either the log of TF, or maximum TF moderation, therefore no moderation method was used for simplicty.

TF-IDF for feature selection[1] involves selecting all features with a TF-IDF value greater than a certain threshold value [3], where:

$$TF\text{-}IDF_{(t,d)} = TF_{(t,d)} \times \log \frac{|D|}{DF_t}$$

As shown in Figure 1, testing was carried out using a range of 0.0-7.0 and an interval of 0.1, to get the best values for the thresholds on both 3-class and 5-class. The range used was deduced through manual inspection.

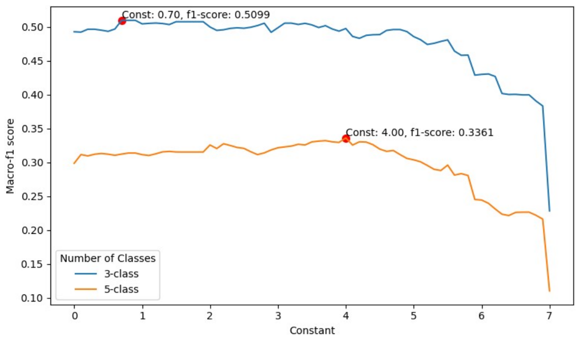

Figure 1: Graph showing macro-f1 score for 3-class and 5-class TF-IDF(selection) against a range of threshold constants.

The decision to explore selecting features based on their *part-of-speech* tags, was made due to it's simple, yet still effective approach. Although, similar to TF-IDF, this method considers features independently from each other, but also does not consider the context of the features at all, for example in regards to their relevance across phrases.

Through inspection of the datasets, nouns, verbs, adjectives and adverbs appeared to be good indicators of sentiment. The combination of selecting them all (NVAR), was selected through manual testing of combinations of these feature types being selected or omited. NVAR was implemented using NLTK's [1] *wordnet*.

Chi^2 was selected as it is effective in assessing the independence between features and the target class, and often shows significant improvements when used as the feature selection for a Naive Bayes Classifier [2][4]. However, Chi^2 considers features independently, meaning wider context within a phrase may not be captured using this method. The Chi^2 statistic for a feature is given by:

$$\chi^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Where O_{ij} is the observed frequency for a given feature and class, and E_{ij} is the expected frequency for a given feature and class.

My own implementation, sklearn's[2] *CountVectorizer* and *TfidfVectorizer* were all tested in conjunction with sklearn's *SelectKBest* and *chi2*, with CountVectorizer yielding the best macro-f1 scores. This is in line with what would be

[1]Rather than for feature extraction.
[2]https://scikit-learn.org/stable/

expected, as Chi^2 makes assumptions that conflict with the assumptions TF-IDF makes. The independence of features assumed by Chi^2 contradicts the weighting from TF-IDF. Chi^2 selects features based on their association with the target sentiment, while TF-IDF weighting prioritises features that are important across phrases, with no regard to sentiment. Sklearn was utilised for this as it offers very efficient and optimized implementations of *CountVectorizer*, *SelectKBest* and *chi2*.

To find the most optimal k values for *SelectKBest*, testing over a range of 0-2000 with an interval of 50 was carried out.

The initial results of the 3 feature selection methods, using only lowercasing as preprocessing, are outlined in Table 1.

Feature Selection	**TF-IDF**	**NVAR**	**Chi^2**
Preprocessing	LC	LC	LC
3-class (features)	0.509871	0.510401	0.507740
5-class (features)	0.336089	0.313913	0.337628

Table 1: Table showing macro-f1 scores of each feature selection technique with lowercasing as preprocessing..

2.1 Optimal Preprocessing

For all three methods, for both 3-class and 5-class, the optimal combination of preprocessing methods was found through similar empirical testing to how the previous constants and parameters where found. Table 2 shows these combinations, and Table 3 shows the macro-f1 results, when the optimal constant(s) and preprocessing for each method are used.

Feature Selection	**3-class**	**5-class**
TF-IDF	ABR, SL, NEG, BIN, LEM + STEM	LC, SL, NEG + BIN
NVAR	LC, SL, BIN + STEM	LC, ABR, SL, BIN + STEM
Chi^2	LC, ABR, NEG + LEM	LC + LEM

Table 2: Table showing the optimal preprocessing combinations for 3-class and 5-class for each feature selection technique.

Feature Selection	**TF-IDF**	**NVAR**	**Chi^2**
Preprocessing	Optimal	Optimal	Optimal
3-class (features)	0.527507	0.515927	0.536483
5-class (features)	0.345227	0.336637	0.359535

Table 3: Table showing macro-f1 scores of each feature selection technique with optimal preprocessing applied.

Based on these results, Chi^2 was selected as the final feature selection method, due to it having the biggest improvement. Optimal preprocessing combinations were found for all_words on both 3-class and 5-class following this.

2.2 Optimal Laplace Parameter

While the Laplace smoothing parameter, α, is typically 1, for smoothing to be applied, the parameter must only be >0. Again, similar to previous testing approaches, an optimal smoothing parameter was found. Table 4 shows the results of Chi^2, with optimal preprocessing, and optimal laplace smoothing parameters[3].

[3]The optimal value found for 3-class was 1.0, hence the scores did not change between Table 3 and Table 4.

	Macro-f1 score	**% increase from initial baseline**	**% increase from majority class baseline**
3-class (features)	0.536483	10.41%	167.62%
5-class (features)	0.369824	30.01%	322.66%
3-class (all words)	0.527507	8.56%	163.14%
5-class (all words)	0.345890	21.60%	295.30%
Average	0.444926	17.65%	237.18%

Table 4: Table showing macro-f1 scores of each feature selection technique with optimal preprocessing and optimal laplace smoothing parameters.

3 Results and Discussion

As evidenced in Table 4, the final models have an average improvement of +17.65% to the baseline macro-f1 results observed for the implemented classifier without any preprocessing or feature selection, and an average improvement of +237.18% on the majority class baseline macro-f1 results. While the 3-class macro-f1 results are greater than those of 5-class, the overall improvement on 5-class is ≈1.9x greater than the overall improvement on 3-class (from the majority class baseline).

Comparing the confusion matrices shown in Figure 2 and Figure 3, one of the most apparent observations is that both models increased the amount of predictions being made for the neutral sentiment. Additionally, for 5-class, the amount of predictions being made on the *very negative* and *very positive* classes was also increased. This is more in line with the distribution of sentiments of the actual development data, than the initial baseline models. Furthermore, looking solely at Figure 3, the model using Chi^2 feature selection performed better in this regard than the model using all words as features.

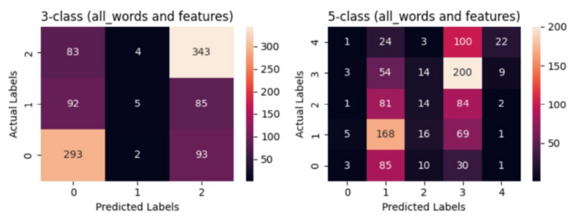

Figure 2: Confusion matrices of the initial baseline.

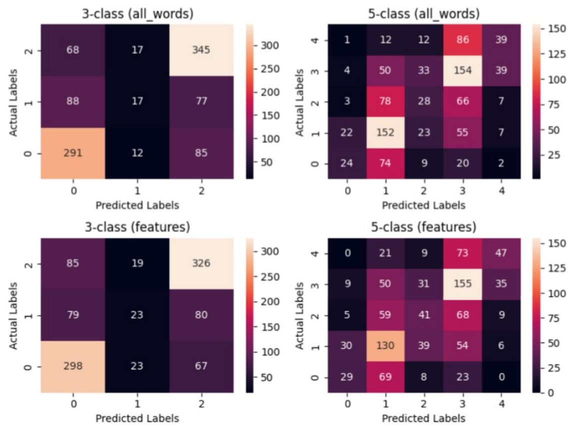

Figure 3: Confusion matrices of the final model.

Looking closer at this, Figure 4 reinforces the conclusion that the model using Chi^2 feature selection performs better than the model using all words as features. For every sentiment, for both 3 and 5 classes, the features model is closer to the actual development data than the all words model.

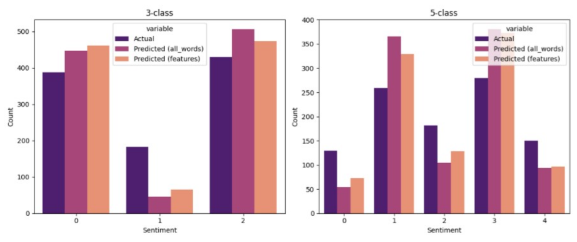

Figure 4: Graph showing number of phrases against sentiment, for the actual development data, and predictions generated from the final models.

3.1 Best Model

In conclusion, the best model is the *'features'* model that uses:

1. Lowercasing and lemmatization preprocessing, with additional negation and replacing abbreviations preprocessing steps for 3 classes.
2. Chi^2 feature extraction, with k constants 1800 and 950 for 3 and 5 class respectively.
3. Laplace smoothing with $\alpha = 1$, $\alpha = 0.87$ for 3 and 5 class respectively.

4 Error analysis

4 examples have been selected for 3 classes to showcase the weaknesses of the final model (Table 5).

Example	SentanceID	Phrase	Actual	Predicted
1	1029	Interesting , but not compelling .	1	2
2	529	A modestly surprising movie .	2	0
3	7118	A less-than-thrilling thriller .	0	2
4	3184	Has all the complexity and realistic human behavior of an episode of General Hospital .	0	2

Table 5: Table showing 4 examples of errors occurring on 3-class.

Example 1 After preprocessing, the features 'not' and 'compelling', become a single feature 'not_compelling'. This feature does not occur in the training data, therefore Chi^2 does not consider it. However 'interesting' (with positive connotations) will be considered, and due to the size of the phrase, this causes the phrase to be incorrectly classified as positive. While the compound feature does not occur in the training data, 'compelling' does, and perhaps an improvement can be made here to handle negations better. If a word is negated, and the negated feature does not occur in the training data, further considerations could be made to locate the non-negated version and continue to consider the feature.

Example 2 This incorrect classification comes from the model not being able to consider features in relation to other features, and rather, all features are considered independently. 'modestly' and 'surprising' may relate to more a negative sentiment independently, however in conjunction, they imply a more neutral sentiment. More complex feature selection methods could be explored, that take into account features in the context of other features, to be able to mitigate better against these types of errors.

Example 3 After preprocessing, the feature 'less-than-thrilling' remains unchanged. This feature never occurs in the training data, so it is not considered by Chi^2. However, 'thriller' occurs 71 times in the training data, and with such a short phrase, the sentiment score comes majorly from this. To improve, a preprocessing method could be introduced to detect and break up dash-separated words into individual features.

Example 4 This misclassification is most likely due to the model not being able to understand that there is negative connotation (namely sarcasm) to 'an episode of General Hospital'. Given the positive nature of other features such as 'complexity' and 'realistic', and that there is no recognition of the negative connotation in the phrase, it has been incorrectly classified as positive.
4 examples have been selected for 5 classes to showcase the weaknesses of the final model (Table 6).

Example	SentanceID	Phrase	Actual	Predicted
1	3025	This is one baaaaaaaaad movie .	0	1
2	3524	A thoroughly engaging , surprisingly touching British comedy .	4	3
3	7230	Worthy of the gong .	1	3
4	4158	I simply ca n't recommend it enough .	3	1

Table 6: Table showing 4 examples of errors occurring on 5-class.

Example 1 After preprocessing, 'baaaaaaaaad' remains the same and it does not exist in the training set, resulting in it not being considered in Chi^2, while being the most relevant feature in the phrase for classification. The model would perhaps need additional, more complex preprocessing to recognise this kind of special language, where 'baaaaaaaad' is actually an intensifier to the adjective bad, rather than a non-english feature. More advanced preprocessing methods could be introduced, to attempt to account for special intensifiers like this - other examples could include fully capitalized words and special punctuation such as '!!!'.

Example 2 This error in classification comes from the model not considering terms in conjunction with one another - 'thoroughly engaging' and 'surprisingly touching' are two very positive phrases. However, with the current model, all 4 features are considered independent, resulting in this prediction of slightly positive rather than positive. To improve upon this, a more complex method of feature selection may be necessary - one that can consider the relationships between features in a phrase.

Example 3 This misclassification may come from the fact that this phrase is most likely a saying, or a reference, perhaps to the movie being reviewed. The model lacks the contextual understanding to interpret this kind of reference. 'gong' appears just twice in the training set as a person's name (both with a sentiment of 3). This causing Chi^2 to value this feature highly, resulting in the misclassification. To improve the model, further experimentation should be carried out to adjust the models sensitivity to infrequent terms, and perhaps this would mean using a more complex feature selection method.

Example 4 This error is a reflection of how well the model handles negation. While a negation preprocessing method was implemented, it was very simple - appending 'not_' to the first and only word that follows the occurrence of the negation word. As demonstrated in this example, appending 'not_' to 'recommend' generates quite a negative feature, yet the phrase is positive. An improvement may be to implement a more complex negation method that would consider a broader context when being applied.

References

[1] Bird, S., and Loper, E. Natural language toolkit (nltk) documentation.

[2] Buslim, N., Putra, A., Wardhani, L., and Busman. Chi-square feature selection effect on naive bayes classifier algorithm performance for sentiment analysis document. pp. 1–7.

[3] Jing, L.-P., Huang, H.-K., and Shi, H.-B. Improved feature selection approach tfidf in text mining. In *Proceedings of 2002 International Conference on Machine Learning and Cybernetics* (2002), vol. 2, IEEE, pp. 944–946 vol.2.

[4] Madasu, A., and Elango, S. Efficient feature selection techniques for sentiment analysis. *Multimedia tools and applications 79*, 9-10 (2020), 6313–6335.