DATA WRANGLING COUREWORK 2

Marcel Troscianko

40173086 SET11521

Algorithms

"Custom" is an algorithm written based on lecture and practical material. "Paper" is an implementation of the Logarithmic Regression approach from "Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach" (Gaydhani et al.)

Two evaluation functions were used:

- Evaluate_model evaluates multiple separate portions of the dataset.
- Train_predict_count evaluates the same dataset multiple times, shuffling it between evaluations.

Data

The algorithms perform on datasets of neutrally offensive, racist and sexist tweets.

Data Pre-Processing

Both projects perform basic pre-processing to make the data usable:

- Each dataset is given labels ("neutral", "racist" and "sexist.")
- The dataset lists are shuffled.
- The datasets are combined (50% of neutral tweets, all sexist/racist tweets).

The research paper approach requires further pre-processing:

- All text is lower-cased.
- Porter Stemmer reduction to normalise all words in the tweets.
- Content deemed "unnecessary" is removed from the tweets:
    - Space Patterns.
    - URLs.
    - Twitter Mentions ("RE @xyz:").
    - Retweet Symbols ("@xyz").
    - Stopwords

Features

Custom

- Word Unigrams, Bigrams and Trigrams.
- N-gram TF-IDF values.

Paper

- Word Unigrams, Bigrams, Trigrams.
- N-gram TF-IDF values.
- L1 and L2 normalized values of the n-grams

Results

Custom

	Min			Max			Average		
	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
Neutral	0.86	0.96	0.9	0.89	0.96	0.91	0.87	0.96	0.909
Racism	0.97	0.94	0.96	0.99	0.95	0.98	0.982	0.944	0.964
Sexism	0.9	0.76	0.82	0.92	0.81	0.84	0.908	0.779	0.831

Figure 1: Custom Algorithm evaluation using train_predict_count

	Min			Max			Average		
	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
Neutral	0.71	0.88	0.82	0.86	0.97	0.89	0.8	0.934	0.86
Racism	0.92	0.76	0.86	0.99	0.93	0.94	0.963	0.849	0.903
Sexism	0.63	0.5	0.61	0.89	0.74	0.76	0.813	0.629	0.705

Figure 2: Custom Algorithm evaluation using evaluate_model

Paper

```
[1/10] Shuffling and Vectorising Data...
[1/10] Performing Grid Search...
[1/10] Best Parameters ::  {'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'newton-cg'}
[1/10] Accuracy ::  0.7908669057160936
[1/10] Elapsed Time ::  0:05:28.268019

[2/10] Shuffling and Vectorising Data...
[2/10] Performing Grid Search...
[2/10] Best Parameters ::  {'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'saga'}
[2/10] Accuracy ::  0.7899177388736554
[2/10] Elapsed Time ::  0:05:23.784656

[3/10] Shuffling and Vectorising Data...
[3/10] Performing Grid Search...
[3/10] Best Parameters ::  {'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'saga'}
[3/10] Accuracy ::  0.7887576460662308
[3/10] Elapsed Time ::  0:05:46.999036

[4/10] Shuffling and Vectorising Data...
[4/10] Performing Grid Search...
[4/10] Best Parameters ::  {'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'saga'}
[4/10] Accuracy ::  0.7966673697532166
[4/10] Elapsed Time ::  0:06:03.437851

[5/10] Shuffling and Vectorising Data...
[5/10] Performing Grid Search...
[5/10] Best Parameters ::  {'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'newton-cg'}
[5/10] Accuracy ::  0.7889685720312171
[5/10] Elapsed Time ::  0:05:45.053306

[6/10] Shuffling and Vectorising Data...
[6/10] Performing Grid Search...
[6/10] Best Parameters ::  {'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'newton-cg'}
[6/10] Accuracy ::  0.7902341278211348
[6/10] Elapsed Time ::  0:05:42.452849

[7/10] Shuffling and Vectorising Data...
[7/10] Performing Grid Search...
[7/10] Best Parameters ::  {'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'saga'}
[7/10] Accuracy ::  0.7912887576460662
[7/10] Elapsed Time ::  0:05:37.579029

[8/10] Shuffling and Vectorising Data...
[8/10] Performing Grid Search...
[8/10] Best Parameters ::  {'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'lbfgs'}
[8/10] Accuracy ::  0.7918160725585319
[8/10] Elapsed Time ::  0:05:47.287234

[9/10] Shuffling and Vectorising Data...
[9/10] Performing Grid Search...
[9/10] Best Parameters ::  {'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'lbfgs'}
[9/10] Accuracy ::  0.7896013499261759
[9/10] Elapsed Time ::  0:05:56.660759

[10/10] Shuffling and Vectorising Data...
[10/10] Performing Grid Search...
[10/10] Best Parameters ::  {'class_weight': 'balanced', 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'saga'}
[10/10] Accuracy ::  0.7868593123813541
[10/10] Elapsed Time ::  0:05:35.625311
```

Figure 3: Literature Algorithm 10-fold Grid-Search Results

	Min			Max			Average		
	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
Neutral	0.66	0.67	0.7	0.78	0.83	0.79	0.735	0.771	0.751
Racism	0.62	0.66	0.71	0.91	0.88	0.86	0.796	0.768	0.779
Sexism	0.61	0.49	0.55	0.8	0.77	0.74	0.677	0.636	0.653

Figure 4: Literature Algorithm evaluation using train_predict_count

	Min			Max			Average		
	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
Neutral	0.81	0.74	0.78	0.86	0.81	0.81	0.83	0.77	0.798
Racism	0.73	0.86	0.79	0.79	0.9	0.84	0.76	0.882	0.816
Sexism	0.73	0.73	0.74	0.78	0.83	0.78	0.754	0.772	0.763

Figure 5: Literature Algorithm evaluation using evaluate_model

Critical Evaluation

Using two evaluation functions was a good idea – the evaluate_model function ensures data isn't cross-contaminated but suffers from the low dataset size while the "train_predict_count" function circumvents the dataset size problem but suffers from the similarity of its corpus across runs. The new function gives consistently better and less dispersed results; the data is shuffled each run and only half the "neither" data is used (reducing data repetition between runs), this is a sign that a higher training corpus results in a better-trained algorithm.

The writers' suggestion to use grid search is interesting, but a bad use of time. "Saga" proved a better solver, but still doesn't dominate the results. This means grid search must be conducted many times to ensure an optimal result is found – a process which can take a very long time for little gain Accuracy gain (Figure 3).

The writers' algorithm gained consistently worse results despite the extra pre-processing. It's likely the removal of stopwords and "unnecessary" content got rid of important data; retweets/mentions can include known controversial figures, URLs could have linked to hateful websites, and stopwords like "him" or "her" in high-sized n-grams could provide extra context insight.

I haven't researched all parameters when first writing my project. It's clear that there are "optimal" classifier parameters for this dataset which I should have implemented. Furthermore, I haven't at all investigated dataset sanitization like the writers have. Removing completely passive terms could improve both the speed and accuracy of the algorithm.

Finally, a known lexicon could have been used to pre-determine the meaning of well-known offensive terminology, phrases and contexts.

Future Work

- Use an entirely different ML library like Keras and compare its efficiency to the current algorithm.
- Acquire a larger dataset to better test portion-based evaluation.
- Split the dataset three ways; Use A% to train the data, B% to conduct a grid-search for parameters, C% to test the data.