Deliverable 3

Connie Chen, Mickey Shi, Nathaniel Imel, Sadaf Khan
{conchen, nimel, bigmigs, sadafkha}@uw.edu

https://github.com/mickeyshi/573-affect-research-group

Abstract

Sarcasm detection is an affect-related natural language processing task of particular relevance for processing textual data from social media platforms. In this paper, we report our results on one sarcasm detection task from iSarcasmEval 2022.

We tranform annotated, pre-processed training tweets from the iSarcasmEval dataset into vectors by using word embeddings generated from the DeepMoji model. We trained various classifiers on these vectors to predict the presence of sarcasm. A subset of classifiers were trained on the same vectors that were supplemented with a polarity feature.

The highest F1-score on the test set we observed was achieved by an ensemble voting classifier (consisting of AdaBoost, MLP Neural Net, and KNN classifiers) that trained on the polarity-enriched vectors. This score (0.6531) was markedly improved in comparison to an "always predict false" and "predict randomly" baselines (0.4615 and 0.4190 respectively). The next best classifiers were the component parts of the Voting model, including variations that were trained with the polarity enriched vectors.

1 Introduction

Sarcasm can be difficult to sense among speakers, even if in-person interaction provides a bevy of accompanying prosodic, facial, and otherwise contextual cues. Predictably, automating the detection of sarcasm in decontextualized text is a complicated and demanding task. SemEval 2022 includes a set of sarcasm detection challenges, under the title iSarcasmEval. The authors of this paper have drawn from iSarcasmEval's task set, with a primary task of binary classification, where tweets are classified as sarcastic or not, and an adaptation task of binary classification, where pairs of tweets will be assigned a sarcastic and non-sarcastic label. For D2, we examined whether Random Forest and Support Vector Machines trained on DeepMoji vectors perform competitively on this primary task. For D3, we explored the performance of additional models, preprocessed iSarcasmEval task data, and tested one method of feature engineering sentiment polarity for sarcasm prediction.

2 Task Description

2.1 Primary and Adaptation Tasks

The primary task of a given model in this paper is to determine whether a given tweet is sarcastic or not. The evaluation metric for this task will be the F1 score for the sarcastic class. Later, an adaptation task will be to discriminate a sarcastic tweet from a counterpart that is a non-sarcastic rephrase; that is, given two tweets about the same event, where one is sarcastic, and the other is not, determine which tweet is sarcastic. A simple accuracy score will be used as an evaluation metric for the adaptation task. Both metrics are in line with the iSarcasmEval metrics.

2.2 Datasets

The training dataset is composed of English language tweets, which are marked for being 'sarcastic' or not. 'Sarcastic' tweets are further annotated for the 'type' of sarcasm they are (*rephrase, sarcasm, irony, satire, understatement, overstatement, rhetorical question*) in addition to author-generated non-sarcastic 'translations' of the tweet. For example, if a sarcastic tweet reads *The only thing I got from college is a caffeine addiction*, the author translation is *College is really difficult, expensive, tiring, and I often question if a degree is worth the stress.* The training dataset has 3466 tweets, 865 of which are marked as 'sarcastic.' There are two testing datasets that correspond to the different tasks (primary and adaptation). The primary dataset has 1400 tweets, 200 of which are sarcastic. The adaptation dataset has 200 tweets (all sarcastic), and 200 rephrases of said tweets.

2.3 Resources

Bamman and Smith (2015) provides a methodology for detection of sarcasm within conversational contexts. Joshi et al. (2017) aggregates generally used approaches for sarcasm detection used across a number of studies. These approaches will be examined and applied for D3.

The description for the shared task can be found at the SemEval 2022 Google site[1], while the GitHub repository hosting the training and test data can be found at Abu Farha et al. (2022a) [2]

Abu Farha et al. (2022b) describes the task, but an affiliated paper has yet to be written. It should eventually be published upon completion of submissions.

3 System Overview

3.1 Modeling

For D2, we focused on exploring a handful of statistical ML classifiers as candidates for our final system model. For D3, we explored additional classifiers, preprocessed our data, and evaluated one approach to feature engineering. As with D2, the system consists of a minimal architecture in which we train a single classifier on the pre-trained embeddings of an existing model. Figure 1 schematically illustrates one of our candidate models.

3.2 Structure of the codebase

The system for D3 targets the primary task only.

The codebase is designed to run the same binary classification (of tweets as sarcastic or not) on multiple possible models and compare their results all at once. We treat the specification of hyperparameters for one model as equivalent to specifying different models. Hyperparameters for models are encoded in YAML files that provide on-the-fly configuration capabilities across test runs.

[1]https://sites.google.com/view/semeval2022-isarcasmeval
[2]https://github.com/iabufarha/iSarcasmEval

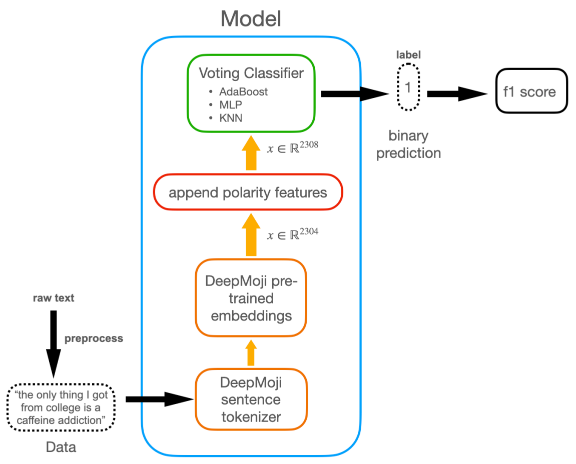

Model
Voting Classifier
• AdaBoost
• MLP
• KNN
$x \in \mathbb{R}^{2308}$
append polarity features
$x \in \mathbb{R}^{2304}$
DeepMoji pre-trained embeddings
DeepMoji sentence tokenizer
raw text
preprocess
"the only thing I got from college is a caffeine addiction"
Data
label
1
binary prediction
f1 score

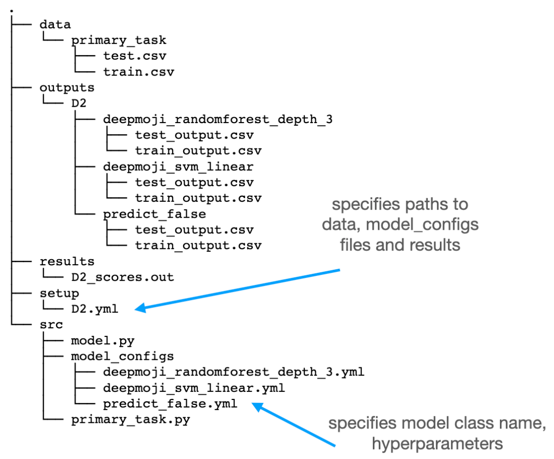
Figure 1: Model architecture for DeepMojiWithRandomForest as a part of system pipeline.

The main project script `primary_task.sh` takes a single argument: the path to a setup file where the paths for data, particular models (and their configs), outputs and results are specified. Each model config file must specify a string name identical to the name of a class implemented in `model.py`. This process is illustrated in Figure 2.

3.3 Caching of vectors

To minimize runtime and remove duplicate work, we cache both the sentence vectors generated by the DeepMoji tokenizer, as well as the DeepMoji vectors with + polarity features added. We use pickle to save these lists into files that persist between runtimes. The main shell script removes these files after running the system, which is necessary if we want to generate a new, clean set of vector files.

Runtimes for certain classifiers were greater than succeeding classifiers using the same cached datasets because time taken to load the saved DeepMoji vectors was factored into the runtime.



```
.
├── data
│   └── primary_task
│       ├── test.csv
│       └── train.csv
├── outputs
│   └── D2
│       ├── deepmoji_randomforest_depth_3
│       │   ├── test_output.csv
│       │   └── train_output.csv
│       ├── deepmoji_svm_linear
│       │   ├── test_output.csv
│       │   └── train_output.csv
│       └── predict_false
│           ├── test_output.csv
│           └── train_output.csv
├── results
│   └── D2_scores.out
├── setup
│   └── D2.yml
└── src
    ├── model.py
    ├── model_configs
    │   ├── deepmoji_randomforest_depth_3.yml
    │   ├── deepmoji_svm_linear.yml
    │   └── predict_false.yml
    └── primary_task.py
```

specifies paths to data, model_configs files and results
specifies model class name, hyperparameters

Figure 2: System architecture (folder structure) for evaluating multiple models.

4 Approach

4.1 Steps from D2 to D3

In D2, we focused on obtaining sarcasm-relevant vector representations from DeepMoji, and finding a simple auxiliary classifer on which to train these representations with non-trivial performance. To provide some sense of measuring performance, we record the performance of a model that mechanically predicts *not sarcastic* for every example. We also record the performance of a model that randomly chooses a class with probability 0.5. These models achieve a test F1 score of 0.46 and 0.42 respectively.

Based on the poor performance of most of the classifiers (many scored no better than the baselines mentioned above) explored in D2, we decided to evaluate several new classifiers, including ensembles. In addition, we preprocess our data, by removing Twitter usernames and web links.

4.2 Polarity

Lastly, we evaluated an approach to engineering sentiment polarity features inspired by Rajadesingan et al. (2015), in which one proposed method of quantifying sarcasm consists in measuring the maximum difference in sentiment polarity in an utterance. The polarity features were obtained from the VADER sentiment analysis tool (Hutto and Gilbert, 2014). This rule-based tool yields a sentence vector containing scores for the features *positive*, *negative*, *compound*, and *neutral*.[3] Essentially, scores for positivity and negativity roughly correspond to the fraction of words in each sentence that are marked positive/negative (taking into account certain, while the compound provides an overall measure of positivity/negativity for the entire sentence, from -1.0 (most negative) to 1.0 (most positive.)

We append scores for positivity, negativity, and compound to the sentence embedding vector generated by the DeepMoji tokenizer, and evaluate these vectors on classifiers separate from classifiers run on the original DeepMoji vector.

4.3 Models

The base models we report here include:

- **Voting**. This is an ensemble classifier of one AdaBoost, MLP, and KNN classifier each.
- **MLP**. This is a feedforward neural network, with an $L2$ penalty of `1e-5`, trained for 1000 epochs and optimized with Adam.
- **KNN**. This is a k-Nearest Neighbors classifier with $k = 5$.
- **AdaBoost**. This is an AdaBoost ensemble classifier that fits at most 50 DecisionTree classifiers.

As in D2, all of our models are built from classifiers that take DeepMoji pre-trained embeddings as input.[4] We report six models: the three models listed above, with and without polarity features appended to their input embeddings. All models are set to 'balance' their labels, to prevent the unbalanced distribution in our primary task data from biasing the classifiers.

5 Results

All of our models for D3 have an F1-score of at least 0.57, which exceeds our D2's best-performing model (linear SVM, F1 = 0.5371). On average, our training times increased from

[3]A detailed description of these features can be found at the public Github repo for this tool: https://github.com/cjhutto/vaderSentiment#about-the-scoring.

[4]For more detail on DeepMoji, see our D2 report.

the classifiers used in D2. Results and runtimes from our highest-performing classifiers are summarized in Table 1 for D3 and Table 2 for D2.

5.1 Revised System Results

The ensemble Voting classifier (consisting of the MLP Neural Net, AdaBoost, and KNN) trained on the polarity-enriched vectors produces the highest F1 score for the test set. The same model trained on non-polarity-enriched vectors performs the next best.

The individual classifiers the Voting ensemble contains are the next best-performing models. While training on polarity-enriched vectors results in better performance for the Voting and KNN classifiers, it seems to disadvantage the MLP Neural Net and AdaBoost models.

In contrast to D2, where most of the models trained are only slightly better than the PredictFalse baseline for the test set (with the non-linear SVM models performing noticeably worse), the models reported in D3 all performed at least 0.1 point better than the PredictFalse and PredictRandom test baselines.

5.2 Runtime

Changes to our approach resulted in increased training time even for the exact same classifiers of SVMs and Random Forests reported on D2 (unreported here). Due to the caching technique utilized for tweet-vector look-up, runtimes can vary widely depending on whether the tweets are being evaluated for the first time or accessed via look-up. Since KNN + Polarity was the first model to cache polarity-enriched vectors, its runtime is multitudes larger than KNN on its own, or other models that train on the polarity-enriched vectors. Hence, runtime is not necessarily a direct measure of the classifier's training and classifying phase timess.

6 Qualitative analysis

We find that the iSarcasmEval test dataset is distinctly different from the training dataset, likely contributing to the distinct difference in performance for F1-scores of the MLP Neural Net between the dev and test sets. The test dataset has far fewer @username mentions - likely sanitized to avoid potentially disclosing personal information.

In general, the dataset was curated poorly. Devoid of context (including previous and following tweets, media, etc), many tweets marked 'sarcastic' would have been difficult to gauge as sarcastic even for a human reader. Some examples of tweets marked as sarcastic include:

- i hate it here ❤ ❤ ❤ ❤ ❤
- Biden is a great President like none other we have had
- Weathers wonderful today! 🎉 🎉 🎉
- You look fantastic in that new dress. It shows off your figure.

Removed from context, including the weather, mood, and political lean of the tweeter, even a human reader would have a difficult time distinguishing between a genuine (if perhaps overpassionate) tweet, and a sarcastic one. Training a classifier to identify patterns between the two risks capturing trends that don't exist within a human attempting to distinguish between a sarcastic interpretation or not.

The authors of the shared task claim sarcasm was self-reported, purportedly removing the problem of ambiguity when compared to a consensus-of-annotators approach. However, removing recipient interpretation arguably omits inherently social (pragmatic) features inherent to some sarcasm.

In addition to these weightier problems of subjectivity, our dataset also suffers from repeats. "I see all the reasons Mark has put forward and they are well explained but you need to ask yourself what is he looking at getting out of this ??,0" appears twice in the test dataset, and there are around 10 such repeated tweets in that dataset alone.

7 Discussion

For D2, we looked into utilizing DeepMoji to produce word embeddings. These word embeddings were used to create vectors for each tweet which were then trained and tested on multiple classifiers. The Linear SVM model performed the best with an F1 score of 0.5371 on the test data set. This performance is slightly better than our D2 baseline PredictFalse model whose F1 score was 0.4615 on the test data set.

Classifier	Dev F1-score	Test F1-score	Train Runtime (s)	Total Runtime (s)
PredictFalse	0.4290	0.4615	0.0202	20.3079
PredictRandom	0.4820	0.4190	0.0801	20.1638
Voting + Polarity	0.7906	0.6531	305.2622	329.099
Voting	0.7660	0.6421	263.4716	286.0384
KNN + Polarity	0.6495	0.6375	1034.3261	1057.2358
KNN	0.6365	0.6194	100.5498	120.787
AdaBoost	0.6657	0.6160	96.262	116.3801
AdaBoost + Polarity	0.6657	0.6160	102.8145	126.6041
MLP Neural Net	0.9610	0.6008	52.6606	73.0585
MLP Neural Net + Polarity	0.9376	0.5777	230.3165	241.8226

Table 1: Summary of results across D3 classifiers

Classifier	Train F1-score	Test F1-score	Train Runtime (s)	Total Runtime (s)
PredictFalse	0.4286	0.4615	0.0054	0.1315
SVM (RBF)	0.5708	0.4450	111.82	111.90
SVM (Linear)	0.7754	0.5371	81.30	81.39
SVM (Sigmoid)	0.5706	0.4445	111.47	111.56
RF (Depth 3)	0.6401	0.5086	6.99	7.07
RF (Depth 4)	0.6815	0.5179	7.15	7.22
RF (Depth 5)	0.7411	0.5116	7.37	7.45

Table 2: Summary of D2 results

		Predicted	
		non-sarc.	sarc.
True	non-sarc.	732	468
	sarc.	63	137

Table 3: Confusion matrix for D2 best classifier (Linear SVM) on the test set.

		Predicted	
		non-sarc.	sarc.
True	non-sarc.	1118	82
	sarc.	131	69

Table 4: Confusion matrix for D3 best classifier (Voting + Polarity) on the test set.

For D3, we continued exploring various different models with DeepMoji embeddings. In addition to the Voting with polarity model performing the best, all three of our polarity models performed comparably well. Because Voting and KNN with polarity achieved higher F1 test scores than their counterparts without polarity, accounting for sentiment polarity seems somewhat beneficial towards improving classification accuracy for those models.It would be interesting to perform a robust statistical analysis of this dynamic. The AdaBoost models with and without polarity achieved the same F1 scores for the test data. Taking a closer look at the output files of each, the predictions of these two models are the same for each tweet.

8 Conclusion

Our exploration of the Voting, MLP, KNN, and AdaBoost models using DeepMoji embeddings along with the employment of preprocessing and sentiment polarity features has yielded substantial improvement from our D2 models. It is likely we will be moving forward with these models for the investigation of our adaptation task.

Recall that the adaptation task is a binary discrimination task: given a pair of tweets, one a sarcastic tweet and another a non-sarcastic rephrase, correctly identify the sarcastic one. We plan to retrieve, for a chosen sarcasm detection model, its probability distribution over labels for each pair of tweets. We will choose the model's prediction as the tweet with the highest probability of being sarcastic.

In terms of error analysis, we are also interested in constructing confidence intervals for our model's scores. We have also considered some feature extraction of our misclassified examples (e.g., looking at polarity of the data).

References

Ibrahim Abu Farha, Silviu Oprea, Steven Wilson, and Walid Magdy. 2022a. isarcasmeval dataset. https://github.com/iabufarha/iSarcasmEval.

Ibrahim Abu Farha, Silviu Oprea, Steven Wilson, and Walid Magdy. 2022b. SemEval-2022 Task 6: iSarcasmEval, Intended Sarcasm Detection in English and Arabic. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Association for Computational Linguistics.

David Bamman and Noah Smith. 2015. Contextualized sarcasm detection on twitter. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 9, pages 574–577.

C. Hutto and Eric Gilbert. 2014. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media*, 8(1):216–225.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)*, 50(5):1–22.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, page 97–106, New York, NY, USA. Association for Computing Machinery.