Podlab at SemEval-2019 Task 3: The Importance of Being Shallow

Andrew Nguyen[1,2], **Tobin South**[1,2], **Nigel G. Bean**[1,2], **Jonathan Tuke**[1,2] and **Lewis Mitchell**[1,2,3]
[1]ARC Centre of Excellence in Mathematical and Statistical Frontiers
[2]School of Mathematical Sciences, The University of Adelaide
[3]Stream Leader, Data to Decisions CRC

Abstract

This paper describes our linear SVM system for emotion classification from conversational dialogue, entered in SemEval2019 Task 3. We used off-the-shelf tools coupled with feature engineering and parameter tuning to create a simple, interpretable, yet high-performing, classification model. Our system achieves a micro F1 score of 0.7357, which is 92% of the top score for the competition, demonstrating that "shallow" classification approaches can perform well when coupled with detailed feature selection and statistical analysis.

1 Introduction

Sentiment analysis from textual data is a well-studied topic, however the particular problem of inferring the emotional context of text-based *conversations* is relatively little-studied. With the increasing abundance of conversational text-based datasets, particularly from social media, there is increasing interest in studying such datasets across a wide range of domains, from unsolicited opinion polling (Cody et al., 2015) to mental health monitoring (Miner et al., 2016). Highly sophisticated systems for emotion detection on conversational data exist (Gupta et al., 2017), however these may not be appropriate for researchers in other fields, and it is desirable to develop *interpretable* models where the informative features can be readily interpreted and measures of confidence can be assigned to individual predictions.

Our aim in entering SemEval2019 therefore was to build a high-performing model using only off-the-shelf tools and some feature engineering, without resorting to deep neural network architectures or overly sophisticated approaches. This model only uses linear Support Vector Machines (SVMs) (Hastie et al., 2009) and feature engineering using the Natural Language Toolkit (NLTK) (Bird et al., 2009). These relatively "shallow" (as opposed to "deep") approaches to classification from text have been used, for example, in event prediction from social media data (Tuke et al., 2018). Despite its simplicity, our system performed very well, achieving a maximum score in the top 22% of all submissions.

2 Task

The task for Semeval 2019 Task 3 is classifying the correct emotion (`happy`, `angry`, `sad` or `others`) on a 3-turn text communication dialogue (Chatterjee et al., 2019). Table 1 shows an example conversation.

Turn 1	You are funny
Turn 2	LOL I know that :p
Turn 3	=)
Label	Happy

Table 1: A example of the 3-turn text dialogue.

The Training dataset contained 5 columns:

- ID: a unique number to identify each training sample;
- Turn 1: the first turn in the 3-turn conversation, written by User A;
- Turn 2: a reply to Turn 1 by User B;
- Turn 3: reply to Turn 2, written again by User A;
- Label: the human-judged label of emotion of Turn 3 based on the conversation for the given training sample. It is always one of the four labels: `happy`, `sad`, `angry` and `others`.

The organisers provided three datasets:

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 292–296
Minneapolis, Minnesota, USA, June 6–7, 2019.

- `Training` contains 30,160 examples with the correct labels and approximately 50% `others`;
- `Dev` contains 2,755 examples with correct labels (approximately 88% `others`);
- `Test` contains 5,509 examples without labels, but participants are told that the split is approximately the same as the `Dev` set, i.e., ≈88% `others`.

This last piece of information given by the organisers regarding the balance of the `Test` set will be critial to our system described below, as we will employ a *post hoc* class balancing technique to match these proportions.

Evaluation is performed using the micro-averaged F1 score for the three emotion classes i.e. `happy`, `sad` and `angry` on the `Test` set.

3 Proposed System

Figure 1 shows an overview of our system, which we describe below in detail.

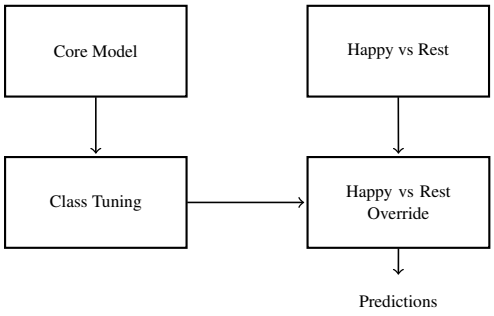

Figure 1: Model pipeline. Our core SVM model performs well for the `sad` and `angry` classes, but underestimates the `happy` class. The "Happy vs Rest" override classifier therefore is designed to achieve high accuracy on this class specifically.

3.1 Feature Engineering

Given a 3-turn text conversation between two users, our system only relied on Turns 1 and 3 while discarding Turn 2 altogether. This was motivated by the insight that the label for each conversation was "the human judged label of *Emotion of Turn 3* based on the conversation" (from the competition website, emphasis ours). This means the overall emotional context of the conversation is dictated by the *reaction* of User A to the chatbot User B's response to A's initial post in Turn 1. Therefore, we hypothesised that the most informative features regarding the emotional context would come from User A's text (with Turn 3 likely to be more informative than Turn 1), and that Turn 2 would largely degrade the impact of these features. This was borne out by experimentation, where inclusion of features from Turn 2 only ever decreased the score compared with models using only data from Turn 1 and Turn 3. Table 2 shows the results of our experiments using different combinations of turns in terms of F1 score.

Turns 1 and 3 were then tokenized using the NLTK Tweet Tokenizer with additional rules[1] to handle symbols, and odd tokens.

These tokens were then scanned for negation words such as {`'not'`, `'isnt'`, `'no'`, `'dont'`} and combined with the next neighbouring token to create one new token. For example, the sentence: `'not happy jane'` tokenized turns into `'not'`, `'happy'`, `'jane'` finally combining negations turns into `'not_happy'`, `'jane'`.

Next the tokens were turn-encoded, meaning each token is prepended with `'1_'` or `'3_'` based on which turn the token was extracted from. For example, tokens from Turn 1 {`'you'`, `'are'`, `'funny'`} are prepended to {`'1_you'`, `'1_are'`, `'1_funny'`}, while Turn 3 tokens {`'=)'`} turn into {`'3_=)'`}.

3.2 Feature selection using TFIDF

The previous section constructs all the tokens found in the `Training`, `Dev` and `Test` data. We then use only the vocabulary found in the intersection of all the datasets. This was done because features which do not appear in the test set have no predictive power on that set. We found in practice that including these features did not improve the scores obtained by our system.

The term frequency inverse document frequency (TFIDF) score was then computed for each term per label basis. The token importance score is computed using the highest TFIDF score for that token across classes minus the second highest. In this manner the importance score used was a measure of a feature's ability to discriminate between classes. Experimentation on the `Dev` set showed that this produced better results than filtering based on, for example, the highest TFIDF score across classes. Figure 2 shows the top 10

[1] https://github.com/andrew-ai/EmoContext2019/blob/master/tokenzie.py

features based on the importance scores.

A cutoff importance score was experimentally selected to further filter down the vocabulary to use only the most informative tokens and avoid overfitting (see Figure 3).

We note that the system is robust to variations in this parameter above a certain threshold – as shown in Figure 3, while the F1 score varies greatly for cutoff values between 0 and 0.005, above 0.005 the F1 score varies only between 0.720 and 0.725.

The features were then one-hot encoded as our final transformation.

Rank	Token	Importance Score	Class
1	3_😂	0.5192	happy
2	3_?	0.2096	others
3	3_😭	0.2033	sad
4	3_haha	0.1823	happy
5	3_i	0.1654	sad
6	3_sad	0.1608	sad
7	3_funny	0.1565	happy
8	1_😂	0.1522	happy
9	3_hate	0.1300	angry
10	3_stupid	0.1224	angry

Figure 2: Top 10 features based on the importance score computed and their associated labels.

3.3 Model pipeline: Core Model

The sklearn Linear Support Vector Classifier (LinearSVC) was used as our classifier. We used a gridsearch strategy with a simple hold out set (`Training` set to train and `Dev` set to validate) to tune the best performing hyper parameters based on F1 score for the `Training` and `Dev` datasets. We used sklearn's default LinearSVC parameters, except for the cost penalty (C) which we tuned and set to $C = 0.3$.

3.4 Model pipeline: Class tuning

Once the core model and model parameters were well tuned, we further tuned the predictions output by the system. Each class of predictions was

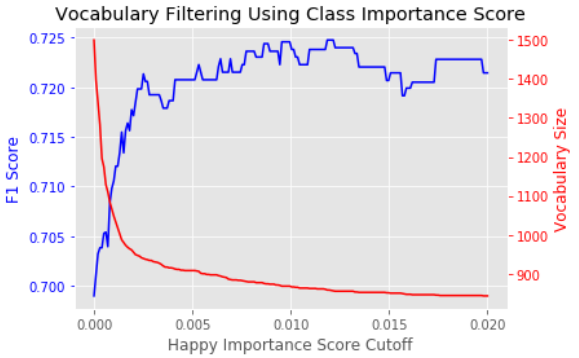

Figure 3: F1 score and vocabulary size as a function of the happy importance score cutoff. The importance scores for the `angry`, `sad` and `others` classes was set to 0.001 from prior tuning. Tuning individually on the `happy` importance score was performed due to it being the weakest classification class and in the end, 0.007 was chosen, which was a local max.

ranked and the least confident predictions based on sklearn's LinearSVC decision_function were shifted to the `others` class. Shifting the classes to have `angry` $\approx 5\%$, `happy` $\approx 4.5\%$ and `sad` $\approx 4\%$ maximised our F1 score. The results of these experiments are shown in Table 2. We note that this relies on the observation that the `Dev` and `Test` sets have approximately the same class-wise distributions, which of course may not hold true if the model is to be used in other contexts.

3.5 Model pipeline: Happy vs Rest override

This model tended to under-predict the `happy` class in terms of precision and recall, therefore an additional model was built to boost classifications within this class. A Happy vs Rest classifier was trained, using the same LinearSVC model and parameters but this time trained on the Training data relabeled as *'happy'* or *'rest'* (`angry`, `sad`, `others`).

After the Core Model predictions and the class tuning, the Happy vs Rest predictions were compared and the most confident predictions based on a confidence threshold was used to override the Core Models predictions. This threshold was chosen by ranking the predictions from the Happy vs Rest classifier in decreasing order of confidence, and converting the top n most confident predictions to `happy` over this threshold. In this case, all predictions from the Happy vs Rest linearSVC decision function with a signed distance over -0.6 were converted. Here the threshold was chosen by inspection of where the predictions appeared to

change from being correct to dubious.

This allowed a more focused model to boost `happy` predictions that the Core Model had missed.

4 Results

Table 2 shows the results of all the system variants described in Section 3. The best overall micro F1 score achieved by our system was 0.735719, or 92.4% of the overall top score in the competition. This placed us in the top 22% of all submissions, and resulted in a final ranking of 37th place.

We note that for each variant of the model, using only Turn 1 + Turn 3 consistently improved the F1 score over using all turns. Adding the Happy vs Rest classifier boosted the `happy` predictions from: Precision = 0.7490, Recall = 0.6514, F1 = 0.6968 to Precision = 0.7368, Recall = 0.6901, F1 = 0.7127. This boosted our overall micro F1 score from 0.729352 to 0.735719. While not shown in Table 2, our *post hoc* class balancing of the final predictions to match the proportions in the `Dev` set was consistently also beneficial in improving our model, as shown in Figure 3.

	T123	T13
NCB	0.7155	0.7188
CB	0.7255	0.7294
NCB + HvR	0.7175	0.7198
CB + HvR	0.7294	**0.7357**

Table 2: Experimental results for system variants where: *T123* = Turn 1 + Turn 2 + Turn 3, *T13* = Turn 1 + Turn 3, *NCB* = No class balancing, CB = Class balancing against test set, *HvR* = With Happy vs Rest override. Note that in each case using using T13 improves over using T123.

Figure 4 shows a confusion matrix for predictions made by our system. In general, the worst misclassifications made by our system were predictions of the `others` class when the correct label was one of the other emotions (or vice versa), rather than misclassifications between emotions, which were negligible.

Figure 5 shows the top 5 "mistakes" – where our system was highly confident in its prediction, but turned out to be wrong. All of these top mislabels were when we incorrectly predicted an emotional class (`angry`, `happy`, `sad`) rather than `others`, and generally when the text contained emojis.

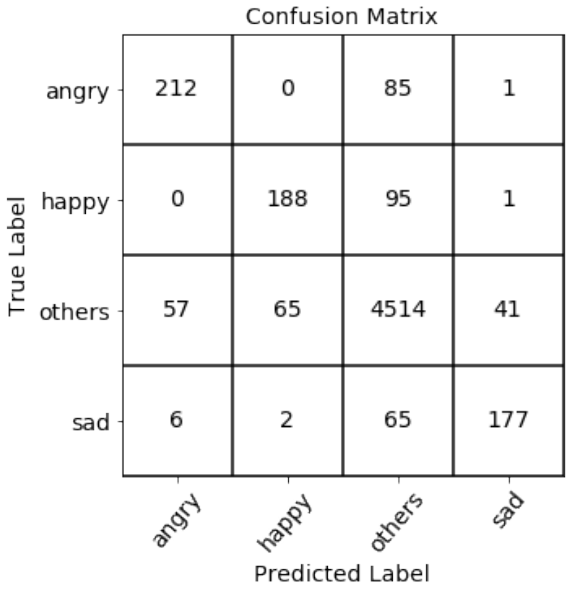

Figure 4: Confusion matrix of misclassifications by our system.

	turn1	turn2	turn3	pred_label	true_label
2198	I didn't get the joke 😂😅	what joke?	You said it was gonna be funny	happy	others
615	But why? funny	I find it funny 😊😤	I am looking funny	happy	others
4853	No one's brain at least	There was a knock on the change room door	I didn't lock it, left it open so no one needs...	sad	others
5284	😂😂😂😂 I should have talked to you earlier	I forget you're a model 😂 just work it off lat...	😂😂😂😂	happy	others
606	😂😂😂😂 😂 thanks for compliment	😑 for what?	Smile😂😂	happy	others

Figure 5: Top misclassifications by our system. All errors were when the correct label was "others" but our system predicted an emotion.

5 Conclusion

That our system performed so well using such a simple approach was very satisfying, and demonstrates that high-quality emotion detection is achievable by practitioners across a range of disciplines. Future work using this system will investigate a longitudinal study of long-term changes in the emotional context of conversations conducted over online social network platforms.

References

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Emily M Cody, Andrew J Reagan, Lewis Mitchell, Peter Sheridan Dodds, and Christopher M Danforth. 2015. Climate change Sentiment on Twitter: An unsolicited public opinion poll. *PLoS ONE*, 10(8):e0136092.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A Sentiment-and-Semantics-Based Approach for Emotion Detection in Textual Conversations. In *Proceedings ofNeu-IR 2017 SIGIR Workshop on Neural Information Retrieval, Shinjuku, Tokyo, Japan, August 11, 2017 (Neu-IR '17)*, pages 1–6.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Adam S Miner, Arnold Milstein, Stephen Schueller, Roshini Hegde, Christina Mangurian, and Eleni Linos. 2016. Smartphone-based conversational agents and responses to questions about mental health, interpersonal violence, and physical health. *JAMA Internal Medicine*, 176(5):619–625.

Jonathan Tuke, Andrew Nguyen, Mehwish Nasim, Drew Mellor, Asanga Wickramasinghe, Nigel Bean, and Lewis Mitchell. 2018. Pachinko Prediction: A Bayesian method for event prediction from social media data. *arXiv preprint: 1809.08427*.