BrainT at IEST 2018: Fine-tuning Multiclass Perceptron For Implicit Emotion Classification

Vachagan Gratian
Universität Stuttgart
vgratian@utopianlab.am

Marina Haid
Universität Stuttgart
haidmarina@gmail.com

Abstract

We present *BrainT*, a multi-class, averaged perceptron tested on implicit emotion prediction of tweets. We show that the dataset is linearly separable and explore ways in fine-tuning the baseline classifier. Our results indicate that the bag-of-words features benefit the model moderately and prediction can be improved with bigrams, trigrams, *skip-one-* tetragrams and POS-tags. Furthermore, we find preprocessing of the n-grams, including stemming, lowercasing, stopword filtering, emoji and emoticon conversion generally not useful. The model is trained on an annotated corpus of 153,383 tweets and predictions on the test data were submitted to the WASSA-2018 Implicit Emotion Shared Task. BrainT[1] attained a Macro F-score of 0.63.

1 Introduction

Our task is to predict emotions of tweets in a dataset where words explicitly mentioning the emotion are masked (Figure 1 and 2). Following the definition of Ekman (1992), there are six "basic" emotions, these tweets have the labels joy, fear, surprise, disgust, anger or sadness. As the model has no access to the *explicit* emotion word, it has to detect it from its *implicit* context, i.e. the situational or causal description of the event. This aspect of the task makes it comparable to centre word prediction from context words.

Twitter language distinguishes itself by a heterogeneous variety of internet vernaculars, abundance of abbreviations, emojis, hashtags and deviation from conventional spelling, grammar, syntax and lexicon. This makes recognition of emotions intricate even for human readers as evident from the noticeably low inter-annotator agreement reported by Balabantaray et al. (2012) or the

[1]Source code is publicly available at: https://github.com/ims-teamlab2018/Braint

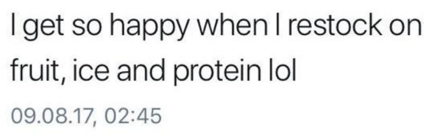
I get so happy when I restock on fruit, ice and protein lol
09.08.17, 02:45

Figure 1: Example of a tweet expressing *joy*.

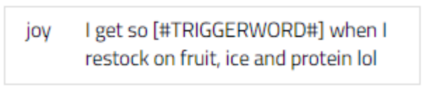
joy I get so [#TRIGGERWORD#] when I restock on fruit, ice and protein lol

Figure 2: The tweet in the dataset: the gold label (left) and the tweet text (right) where the emotion word is masked.

"testing" of the IEST dataset on English native-speakers which resulted in an F-score of 0.45 (Klinger et al., 2018).

2 Related Work

Previous research on sentiment analysis and emotion analysis of Twitter data often disagrees on the benefits or disadvantages of the various approaches, algorithms and feature models.

In Psomakelis et al. (2014) linear and multi-layer classifiers are evaluated on sentiment analysis of tweets and is found that learning-based approaches outperform lexicon-based approaches explaining this chiefly by the lack of contextual information that lexical entries (such as polarity scores) express in the unigram model. Kouloumpis et al. (2011) found a mixed feature set of unigrams and n-grams beneficial for sentiment analysis, but found that adding POS-tags to the feature set drops the model's performance and questioned its usefulness specifically on Twitter data. Aston et al. (2014) observe that the voted perceptron performs quite well using only character n-grams and propose a feature-reduction method that dramatically decreases runtime with-

Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 243–247
Brussels, Belgium, October 31, 2018. ©2018 Association for Computational Linguistics
https://doi.org/10.18653/v1/P17

out compromising performance. Conversely, Balabantaray et al. (2012) evaluate a "greedy" feature model, including n-grams, POS-tags, bigram POS-tags, dependency tags, affection labels etc. Interestingly, the authors of this paper add a seventh class, *no emotion* to the six basic emotion classes and find that the multi-class SVM attains a high accuracy score with a panoramic feature model.

3 Methods

3.1 Multi-class Perceptron

We design our model following the "one-against-all" approach described in Allwein et al. (2001) by reducing the multi-class prediction task into $k = 6$ binary classification problems. We add weight vectors for each emotion class $(w^{joy}, w^{fear}, ...)$. Prediction is made by assigning each tweet vector x_i the label that gets the highest confidence:

$$\hat{y} = argmax\ w^y \cdot x$$

$$y \in \{sad, joy, disgust, fear, surprise, anger\}$$

For each incorrect prediction, the model is updated by adding the tweet vector x_i to the true label's weights y_i and subtracting it from all the other weights:

$$if\ \hat{y} \neq y_i :$$

$$w^y \leftarrow w^y + x_i$$

$$w^{\hat{y}} \leftarrow w^{\hat{y}} - x_i$$

After our first experiments we upgraded our model to the averaged perceptron as defined in Collins (2002) and as discussed in Kazama and Torisawa (2007). Doing so, we set the final weights to be the average of all updated weights during training. Additionally, we randomize the order of tweets before each training epoch to reduce overfitting.

3.2 Features

Our feature set consists of unigrams, bigrams, trigrams and what we call *skip-one*-tetragrams. We use a combination of n-grams as our feature set and optionally add POS-tags.

The unigrams are modified depending on the selected preprocessing mode. This can be either *reductive* (surface word is reduced to its stem or lowercased, stop words and punctuation are removed, emojis and emoticons are replaced by labels, numbers are replaced by $\langle NUM \rangle$ tag) or *additive* in which case stems, labels and tags are added to the feature set alongside the surface form. Bigrams and trigrams are added to the feature set as they are. Tetragrams are duplicated and respectively the second and the third tokens[2] are replaced with $\langle SKIP \rangle$. We expect that this will generalize phrases that only differ in one token. E.g., "he loves red apples" with *skip-one* is "he loves $\langle SKIP \rangle$ apples" and will match with "he loves green apples" in another tweet.

We calculate the feature values using one of the following measures: binary (0 or 1), count, frequency or tf-idf.

4 Experiments

4.1 Dataset

The dataset we use is provided by the WASSA 2018 Implicit Emotion Shared Task[3]. It is a corpus of 153,383 tweets annotated with distant supervision where each tweet originally contained one of the six emotion words (joy, fear, surprise, disgust, anger, sadness) or their synonyms. These words are masked in the dataset, as are usernames and URLs. The dataset is described in detail in Klinger et al. (2018). We use a test set consisting of 28,757 tweets, provided by the IEST as well.

4.2 Preprocessing

We tokenize and normalize tweets using methods that allow for the orthographic anomalies of tweets (e.g., missing space between words and punctuation marks; use of punctuation marks as emoticons). Tokens are labelled by their type (word, punctuation, numerical, emoji, emoticon, hashtag or URL). Depending on our choice between the reductive or additive modes, word tokens are replaced or complemented with stems, all other types by a label or tag. For example, the emoji 😀 and the emoticon *:))))* both are replaced or complemented by *laughing*[4]. Numbers like e.g. *1948* are replaced or complemented by $\langle NUM \rangle$.

We also add counts of word classes in the tweet using the NLTK[5] part-of-speech tagger. Option-

[2]Doing the same with the first and last tokens would reduce it to a trigram.

[3]Available at: http://implicitemotions.wassa2018.com/data/

[4]We created our own libraries for common emojis and emoticons. For not common emojis we used the Python library *emoji*.

[5]https://www.nltk.org/

Feature vectors	Conv	**Macro F**
Binary	0.8	**0.382**
Count	0.78	**0.412**
Freq	0.57	**0.436**
TF-IDF	0.79	**0.401**

Table 1: Results of testing the baseline with unigrams. $T = 150$.

ally stopwords and punctuation marks can be removed and tokens can be lowercased.

These preprocessing options are only applied to unigrams, since they would otherwise disturb the word order in n-grams.

4.3 Experimental Setup

We evaluate our model on the test data described in section 4.1. We consider Macro F-score as the evaluation metric and calculate Precision and Recall scores for each emotion class. We run our experiments with learning rates ranging from 0.1 to 0.5, but choose for 0.3 in later experiments as the model seems to converge slightly better in this case. For the initial model we set the number of epochs $T = 150$, but with averaging of the weights, $T = 50$ seems reasonable as the learning curve plateaus already after 30-35 epochs. During each epoch we calculate the accuracy of the predictions on the train data (we refer to this measure as *Convergence* or *Conv*).

Additionally, after each epoch the model is evaluated on the test data whereby the weights are not adjusted so the test data remains unseen. With these two measures we can track how the model adapts to the train data in comparison to its performance on the test data.

4.4 Results

We conduct four groups of experiments in increasing complexity of the feature set.

Group 1. First, we test the ”vanilla” perceptron with unigrams and with minimal preprocessing (only tokenization). We try all four vector value calculations, but since frequency attains the highest score, we choose only that one for the next experiments. Results of this group of experiments are shown in table 1.

Group 2. We then update our model to the averaged perceptron and shuffle tweets before each epoch. This raises the F-score from 0.44 to 0.52. Subsequently we evaluate the model with more advanced preprocessing options. Both reductive and additive modes are considered. Results of Group 2 experiments are included in Tables 2 and 3. Since the impact of these options is either negative or positive but negligible, we choose for no unigram preprocessing options in the next experiments.

reductive options	Conv	**Macro F**
none	0.47	**0.519**
replace emoji/emoticon with label	0.47	**0.516**
replace number with tag	0.47	**0.519**
remove stopwords	0.50	**0.481**
remove punctuation	0.48	**0.511**
lowercase	0.46	**0.511**
replace word with stem	0.49	**0.529**
all of the above	0.52	**0.468**

Table 2: Results of *reductive* preprocessing options using unigram frequencies. $T = 50$.

additive options	Conv	**Macro F**
add emoji/emoticon label	0.50	**0.545**
add number tag	0.50	**0.545**
add covercased token	0.50	**0.546**
add stem	0.49	**0.536**
add stem + emoji/emoticon label	0.49	**0.537**
add stem + emoji/emoticon label + number tag	0.50	**0.546**
all of the above	0.49	**0.537**

Table 3: Results of *additive* preprocessing options using unigram frequencies. $T = 50$.

Group 3. In Group 3 of the experiments we incrementally add bigrams, trigrams, skip-one-tetragrams and POS-tags to the feature set (Table 5 and Figure 3).

Group 4. Finally, we repeat the experiments of Group 3 non-incrementally. Table 5 shows the results.

4.5 Discussion

We observe a strong improvement of the averaged perceptron with shuffling over the baseline perceptron. Predictions get better as more n-grams are added to the feature set, which is self-evident as they capture more contextual information. The learning curve converges on the training data after trigrams are added, which indicates that the dataset is linearly separable.

As it was found by Saif et al. (2014), we

Feature vectors	Conv	**Macro F**
Unigram	0.50	**0.546**
Bigram	0.88	**0.607**
Trigram	0.99	**0.616**
Skip-one-Tetragram	1.00	**0.625**
POS-tags	0.99	**0.632**

Table 4: Results of third group of experiments: Feature sets are added incrementally. $T = 50$.

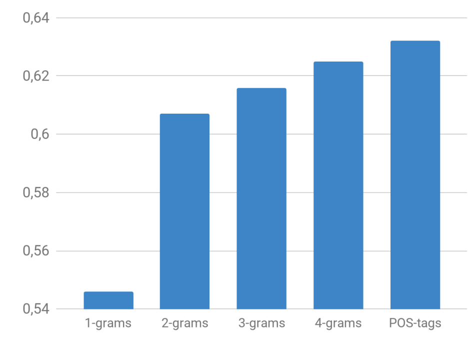

Figure 3: Macro F-scores obtained on different feature sets. Feature sets are added incrementally (i.e. the feature type on the right contains all on its left).

Feature vectors	Conv	**Macro F**
Unigram	0.50	**0.546**
Bigram	0.90	**0.585**
Trigram	0.99	**0.571**
Skip-one-Tetragram	1.00	**0.559**
POS-tags	0.99	**0.301**

Table 5: Group 4 experiments: feature set includes only one n-gram or the POS-tags. $T = 50$.

confirm that classic stopword filtering decreases performance and observe that similarly lowercasing, punctuation removal, stemming and emoji/emoticon conversion have a negative or neutral impact.

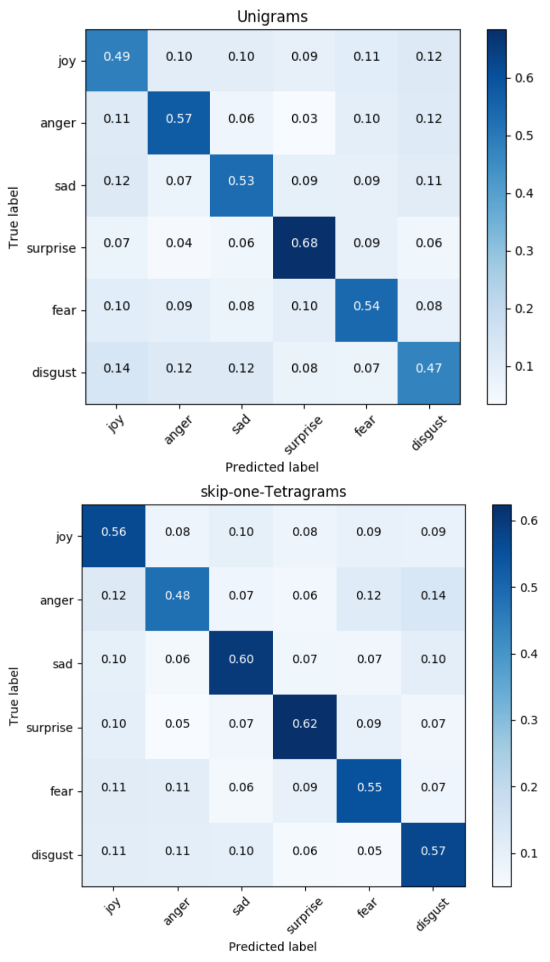

Figure 4: Confusion matrix of two models trained on Unigram and skip-one-Tetragram features.

5 Future Work

The model and approaches described in this paper can be improved in two directions: enhancing the feature set and addressing the limitations of the multi-class perceptron. In the ”one-against-all” model the output of each classifier is treated as a confidence measure, for a more precise prediction this score can be calibrated into probability. As demonstrated in Figure 4, models trained on different feature sets show different strengths and weaknesses in their predictions. This disparities can be exploited by adding ”redundant” classifiers for the same emotion class and train them differently. A final prediction can be made based on a simple majority vote or a distance measure between the individual predictions. As described in Garcia Cifuentes (2009), this can improve the models performance. In this scenario, the preprocessing options described in 4.2 could also prove to be helpful.

We would also like to try other multi-class reduction approaches on the same implicit emotion prediction task, such as ”all-pairs” or ”error-correcting code”, both known to perform better than the ”one-against-all” approach (Allwein et al., 2001).

6 Conclusion

In this paper we evaluate a multiclass averaged perceptron on implicit emotion detection in tweets. We discuss how different preprocessing options and feature sets affect its performance. In particular, we demonstrate that the bag-of-words model enhanced with bigrams, trigrams, skip-one-tetragrams and POS-tags shows strong improvements over the initial baseline. Conversely, stopword filtering, lowercasing, stemming, emoji and emoticon conversion, proved not to be helpful in our experimental settings.

References

Erin L Allwein, Robert E Schapire, and Yoram Singer. 2001. Reducing multiclass to binary: a unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141.

Nathan Aston, Jacob Liddle, and Wei Hu. 2014. Twitter sentiment in data streams with perceptron. *Journal of Computer and Communications*, 2(03):11.

Rakesh C Balabantaray, Mudasir Mohammad, and Nibha Sharma. 2012. Multi-class twitter emotion classification: A new approach. *International Journal of Applied Information Systems*, 4(1):48–53.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Cristina Garcia Cifuentes. 2009. Multi-class classification with machine learning and fusion.

Junichi Kazama and Kentaro Torisawa. 2007. A new perceptron algorithm for sequence labeling with non-local features. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Roman Klinger, Orphée de Clercq, Saif M. Mohammad, and Alexandra Balahur. 2018. Iest: Wassa-2018 implicit emotions shared task. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Brussels, Belgium. Association for Computational Linguistics.

Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11(538-541):164.

Evangelos Psomakelis, Konstantinos Tserpes, Dimosthenis Anagnostopoulos, and Theodora Varvarigou. 2014. Comparing methods for twitter sentiment analysis. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management-Volume 1*, pages 225–232. SCITEPRESS-Science and Technology Publications, Lda.

Hassan Saif, Miriam Fernández, Yulan He, and Harith Alani. 2014. On stopwords, filtering and data sparsity for sentiment analysis of twitter.