Team 'How Does That Make You Feel?': SemEval2016 Task 4 Final Report

Evan Akers – M08637241
Kyle Arens – M06181631
Ryan Benner – M04890768
Joe Hirschfeld – M08704926

NAACL submission

Abstract

This document is a final report for the SemEval2016 Task 4 Sentiment Analysis in Twitter. It is made up of five subtasks, labeled A through E, covering both labeling tweets on a 3-point and 5-point scale as well as quantification and prediction of an entire grouping of Tweets (Preslav Nekov et al., 2016). We approached this project by developing a classifier for the 5-point classification and quantification subtasks, then generalizing this for the 3-point subtasks.

1 Task Description

Twitter provides a convenient platform for us to share information, news, and opinions. As such, it can provide a rich dataset of Tweets about a variety of topics, such as politics, products, and many other areas. This has made it popular for various natural language processing research projects (Preslav Nekov et al., 2016). This paper describes our approach to the SemEval2016 Task 4 challenges.

The SemEval2016 Task 4 are described as follows:

Subtask A: Given a tweet (irrespective of topic) and decide on a 3-point scale whether the tweet conveys a positive, negative, or neutral sentiment.

Subtask B: Subtask B is similar to Subtask A, but takes a given tweet and a topic, classifies a tweet on a binary scale, positive or negative, in regard to the given topic.

Subtask C: The same challenge as Subtask B, but evaluates the tweets on a 5-point scale: {very positive, positive, neutral, negative, very negative}.

Subtask D: Related to Subtask B, this task requires us to estimate the distribution of a set of tweets about a given topic across a binary positive or negative spectrum.

Subtask E: Similar to Subtask D but an extension of Subtask C, this task asks us to estimate the distribution of tweets about a topic across the 5-point scale (described earlier in Subtask C). The official description of the Subtasks may be read on SemEval2016's Task4 website (Preslav Nekov et al., 2016).

Due to the similarities in these subtasks as well as the given data, we will approach Subtask A as a learning task to get familiar with the approaches of sentiment analysis, and then tackle Subtasks B and D together as well as Subtasks C and E together.

2 Approach

Our general approach to solving the problems of Subtasks A through E consists of Preprocessing, Annotating/Feature Abstraction, and processing through a 2-layer Convolutional Neural Net. We used the Tensforflow library for our neural net (Martin Abadi *et al.*, 2015).

Prior research on this task shows various groups utilizing a myriad of features, such as sentiment lexicons, spell checker, and hashtag annotation (Mahmoud Nabil et al., 2016). Other groups performed other forms of preprocessing, such as replacing URLs and usernames with standardized

strings as well as adjusting elongated ("wooooow") words to a standard representation ("wow") (Giovanni Da San Martino et al., 2016).

We both borrow and deviate from these features in our project. Of note, we deviate by focusing almost exclusively on conjunctions and punctuation. We follow similarly to other groups by utilizing POS tagging, a sentiment lexicon, and spell checking/correcting. We will further explore our methodology below.

2.1 Preprocessing

We prepared our data for use in our feature abstraction layer, by substituting all references to Twitter's embedded shortened links (http://t.co…) to "httptco".

Our current preprocessing is vastly minimal compared to other groups (who included various techniques such as separating hashtags into their individual words and expanding abbreviations) we believe there might be relevance in the manner of *how* someone refers to another entity and the abbreviations used that indicate positive, negative, or neutral sentiment (Balikas and Amini, 2016; Räbiger *et al.*, 2016; Yadav, 2016). For example, if a person is happy with Microsoft, they might type Microsoft, utilizing correct capitalization whereas a negative sentiment tweet might ignore proper capitalization or spelling altogether.

After preprocessing we passed our tweets to our tokenization and feature abstraction layer. We assigned the following features to each token in a tweet.

1) The count of each of the following punctuations [".", "..", "…", "?", "!"] are used in the sentence the token is a part of.
2) Whether or not the word in the token is spelled correctly.
    a. After spell checking, we then naively correct the spelling using PyEnchant as a naïve spell-correction tool (Ryan Kelly, 2014**)**.
3) Whether or not the token is a hashtag.
4) The part of speech of the token (Edward Loper and Steven Bird, 2002).
5) The count of coordinating conjunctions associated with a token as well as which side of the conjunction the token appears on (Edward Loper and Steven Bird, 2002).
6) A token-level sentiment score based on its positive score, negative score, and objectivity score (Edward Loper and Steven Bird, 2002).

From this, we then took our tweets and passed them into a two-layer convolutional neural network that we created utilizing the Tensorflow framework to learn to predict sentiment given the above features (Martin Abadi *et al.*, 2015).

Presently, we train each subtask using the appropriate amount of classifiers, as dependent upon the subtask. For subtasks B-E we now use a naïve classifier to determine whether or not the sentiment of a Tweet is directly targeted at the topic, or indirectly, such as through comparison with a superior (or inferior) product, external entity, etc.

After training our model, we used the Test data to see how well our features could predict sentiments and estimate the distribution of positive and negative tweets.

We believe our approach on analyzing tweets based on conjunctions is relatively unique to how teams approach this problem. We are aware that is has been used in some tools, such as VADER but haven't seen much explicitly done over training on tweets with contractions (Hutto and Gilbert, 2015).

3 Results

3.1 Subtask A

By using the F_1^{PN} metric defined by Nakov *et alii* and using their provided script to score our result on the 2016 Subtask A devtest data, we achieved a score of 0.3296 during our initial proposal work. After further work on our system, we used the 2016 Subtask A Test data and achieved a final score of 0.2571. This puts us at Rank 35 (last place) if we seat our scores against the 2016 competition. Note that higher is better (Nakov *et al.,* 2016).

3.2 Subtask B

By using the ρ^{PN} metric defined by Nakov *et alii* and using their provided script to score our result on the 2016 Subtask BD devtest data, we achieved a score of 0.512 during our initial proposal work. After further work on our system, we used the 2016 Subtask BD Test data and achieved a final score of 0.501. This puts us at Rank 20 (last place) if we seat our scores against the 2016 competition. Note that higher is better and baseline should be around 0.5 (Nakov *et al.,* 2016).

3.3 Subtask C

By using the MAE^{M} metric defined by Nakov *et alii* and using their provided script to score our result on the 2016 Subtask CE devtest data, we achieved a score of 1.28 during our initial proposal. After further work on our system, we achieved a final score of 1.40 against the 2016 Subtask CE Test data. This puts us at Rank 11 (out of 12) if we seat our scores against the 2016 competition. Note that lower is better (Nakov *et al.,* 2016).

3.4 Subtask D

By using the KLD metric defined by Nakov *et alii* and using their provided script to score our result on the 2016 Subtask BD devtest data, we achieved a score of 0.408 during our initial proposal. After further work on our system, we tested our data against the 2016 Subtask BD test data, and achieved a final score of 0.726. This puts us at Rank 15 (last place) if we seat our score against the 2016 competition. We do outperform one baseline metric. Here, the lower the score the better (Nakov *et al.,* 2016).

3.5 Subtask E

By using the EMD metric defined by Nakov *et alii* and using their provided script to score our result on the 2016 Subtask CE devtest data, we achieved a score of 0.378 during our initial proposal. After further additions to our approach, we achieved a final score of 0.721 when running against the Subtask CE Test data. This puts us at Rank 10 (out of 11) and we did outperform one baseline measure. Note that lower is better (Nakov *et al.,* 2016).

4 Discussion

One of the trends we noted within our prediction algorithms during our proposal, was that our predictions were frequently optimistic, and generally assigned a positive sentiment to a Tweet. In our final scores, a review of the data indicates that this trend has worsened and our system is much more likely to predict a positive sentiment over anything else. This indicates to us one of two things: (1) that the training data utilized had a higher distribution of positive tweets than any other sentiment, thus requiring our model to have a more diverse training set or (2) that grammatical constructs, punctuation, etc. are fluidly used between tweets of various sentiments and cannot be easily used to predict a sentiment.

Given the results of our project (see section 3), we believe that *how* something is written is significantly less important than *what* is written. The content of a document is much more important to sentiment analysis than the grammatical features of the document. As we added more annotations for grammatical features, the impact of the features addressing what was being said was minimized, causing our results to worsen.

Moving forward, we believe grammar is not an adequate fit for a primary classifier in sentiment analysis, and that it is either better suited to specific edge cases or solutions that utilize a more complex approach, such as VADER (Hutto and Gilbert, 2015). We see a potential application of grammar in detecting sarcasm, for example.

Acknowledgments

Evan Akers wrote the spell-checking feature extraction function.

Kyle Arens developed the data parsing algorithm to break down input data into data structures consumed internally by our algorithms. He extended this into creating the "data flow" pipeline from reading the input files to creating the neural net data objects. This included reading the input data, performing the text preprocessing on that data, and transforming into the output data format. Kyle wrote the word indexing function that took the list of unique

words and the counts from the input set and transformed that into a list of significant words and their corresponding indices for use in our vector graphs consumed by the neural net. Next, he wrote the word to vector function which takes the previously mentioned list of significant words, all of the input tweets, and list of features and abstracts them into three multi-dimensional graphs that match the input data structure the neural net requires. Kyle then wrote the functionality used for naive sentiment tagging in the feature extraction and tokenization portion of the code. Kyle worked with Joe to create the conjunction net (not used in our proof of concept, but will be included for final submission) and verified the mathematical algorithms within the neural net. Kyle improved code-readability by refactoring variable names and documenting all code written. Kyle worked with Ryan to complete and proofread both the slides for the final presentation, and the final report.

Ryan Benner initially contributed at a higher level to this project, by doing background research on semantic analysis and what other groups have done previously for SemEval-2016 Task 4. After this, working with Joe, Ryan came up with the initial preprocessing and target feature extractions we used in our preliminary results. He was directly responsible for most of the feature implementation annotation on tokens, writing the following algorithms for annotation: hashtag annotation, punctuation annotation, conjunction counting and annotation, and part-of-speech annotation. Ryan enhanced the intial spell-checking function provided by Evan to also correct spelling. These functions contributed to the data-flow pipeline. He also contributed to the data pipeline by filtering out "Not Available" tweets. Ryan created the word count function for creating a list of unique words counts, denoted by their token-form and associated part-of-speech. After this, Ryan wrote the project proposal document. For the final report, Ryan worked with Kyle to prepare both the final report and the final presentation. Ryan also obtained the result data for both the initial proposal and final report from running the prediction algorithms and compiled them for use in those reports. Early in the development process, Ryan fixed minor bugs that occurred as we constructed the data-flow pipeline.

Joe Hirschfeld wrote various parts of our code base. First, he authored the scripts to download the data from Twitter and filter it according to the Twitter terms of service to create both algorithm- and score-ready versions of all the data we used for training and dev-testing. Next, he authored our convolutional neural net. Joe took the data parsed and massaged by Kyle and Ryan's code, and transformed them into various vectors and matrices for use by the sentiment engine. During the neural net development process, he worked with Kyle on the math-related algorithms of the neural net and afterward, spent his time tuning the various hyper-parameters on the sentiment neural net. During the development cycle, Joe took time to clean up the code database, doing necessary refactors to keep the code clean and maintainable. He further enhanced the original neural net we used during the proposal by adding an extra layer for sentiment prediction.

References

C.J. Hutto and E.E. Gilbert. 2014. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, Ann Arbor, MI.

Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceeding ETMTNLP '02 Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics – Volume 1,* Stroudsburg, PA, US.

Giovanni Da San Martino, Wei Gao, Fabrizio Sebastiani. 2016. QCRI at SemEval-2016 Task 4: Probabilistic Methods for Binary and Ordinal Quantification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016),* San Diego, US.

Georgios Balikas and Massih-Reza Amini. 2016. TwiSE at SemEval-2016 Task 4: Twitter Sentiment Classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, US.

Mahmoud Nabil, Mohamed Aly, and Amir F. Atiya. 2016. CUFE at SemEval-2016 Task 4: A Gated Recurrent Model for Sentiment Classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016),* San Diego, US.

Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia,

Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Software available from tensorflow.org.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 Task 4: Sentiment Analysis in Twitter. Association for Computational Linguistics. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, US.

Ryan Kelly. PyEnchant: a spellchecking library for Python. 2014. Package available for download from pythonhosted.org/pyenchant.

Stefan Räbiger, Mishal Kazmi, Yücel Saygin, Peter Schüller, and Myra Spiliopoulou. 2016. SteM at SemEval-2016 Task 4: Applying Active Learning to Improve Sentiment Classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, US.

Vikrant Yadav. 2016. thecerealkiller at SemEval-2016 Task 4: Deep Learning based System for Classifying Sentiment of Tweets on Two Point Scale. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, US.