# Crowdsourcing Bias in Sentiment Analysis

Authors: Ben Choi, Ricky Noll
Department of Computer Science, Washington University, St. Louis, Missouri 63130

Advisor/Professor: Chien-Ju Ho

## 1 Author Information

1. Ben Choi (benjaminchoi - 443821)

2. Ricky Noll (rickynoll - 443896)

## 2 Introduction

### 2.1 Original Project Proposal

To recap our original project proposal, we planned on creating a sentiment analysis algorithm that would ultimately separate tweets into one of three categories: positive, neutral, or negative. To do so, we wanted to build off of the work of *Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis* by Wilson, Wiebe, and Hoffmann [1] (henceforth referred to as paper 1) and *Twitter as a Corpus for Sentiment Analysis and Opinion Mining* by Pak and Paroubek [2] (henceforth referred to as paper 2). The two papers take very distinct approaches to creating a sentiment analyzer and we wanted to take concepts from both papers to create our own sentiment analysis algorithm. In the spirit of crowdsourcing, we also wanted to make our algorithm more robust by adding some "human in the loop" adjustments. More specifically, instead of classifying tweets outright, we wanted to run a regression to assign tweets values from a range of -1 to 1, allowing us to more closely examine tweets for which the algorithm predicted incorrectly or just barely predicted correctly.

### 2.2 Project Pivots

Originally, our plan was to use the features described by paper 1 and incorporate the social media speech elements detailed by paper 2 to produce a sentiment analyzer that would nicely reconcile the ideas of the two papers for the purposes of analyzing tweet sentiments. As we continued on, however, we began to realize that the authors of paper 1 only uploaded one of about three lexicons they described using in their paper. Without access to these other dictionaries, it became nearly impossible (without a lot of manual work and tagging, on our part) to implement a sentiment analyzer in the way we originally proposed. Thus, we pivoted to more closely replicate paper 2 while still trying to incorporate elements of the work in paper 1. In doing so, we built a fairly robust sentiment classifier, but one that ran into certain challenges inherent in working with crowdsourced data. Ultimately, our project focuses on these challenges and the various ways one can overcome them (and the pros and cons of these approaches).

## 3 Data

Originally, we had intended on scraping Twitter ourselves, labeling tweets by hand, and then training our algorithm on this labeled dataset. Unfortunately, this ended up being such a huge task for just the two of us that it became infeasible to do without crowdsourcing it through AMT (something we did not sign up to do, in time). As a result, we ended up using a dataset from kaggle, which used crowdsourcing to label positive, neutral, and negative tweets regarding airlines. As the labels were crowdsourced, we do not actually know the gold label (how the tweeter actually felt) for nearly all tweets. However, each label did come with a confidence value ranging from 0 to 1 which shows what percentage of total labelers la-

beled the tweet with its final label. This became the dataset we used for the duration of our project. [3]

The data came with 15 features, including certain features we did not need like timezone and retweet count. We filtered these to just 3: `airline-sentiment`, `airline-sentiment-confidence`, and `text`.

# 4 Data Analysis

## 4.1 Classifier Choice

We ended up using a linear kernel SVM which analyzed a bags of words representation of tweets. Although we focused on the 3-way classification between positive, neutral, and negative, we also experimented with binary classifications. More specifically, we trained and tested a positive-negative classifier and a polar-neutral classifier. The positive-negative classifier simply distinguishes between positive and negative tweets while the polar-neutral classifier distinguishes between polar (either positive or negative) and neutral tweets.

## 4.2 SVM Choice

Research in the field of text analysis (including papers 1 and 2) traditionally use either a Multinomial Naive-Bayes or a SVM to create sentiment analyzers, which led us to try both types of classifiers in our project. We ultimately decided to go with the SVM because it yielded a better testing accuracy in our trials.

## 4.3 Linear Kernel Choice

Once we settled on the choice of a SVM, we then tested a linear kernel, a polynomial kernel, a RBF kernel, and a sigmoidal kernel. Once again, the linear kernel yielded the best results and thus we stuck with it for the rest of our project.

## 4.4 Bags of Words Choice

When making the decision on how to turn the tweet texts into feature vectors, we considered 2-grams, 3-grams, part-of-speech (POS) frequency, and a bags of words representation. Paper 1 explicitly uses 3-grams when building its classifier while paper 2 recommends using 2-grams. Both papers also use POS frequency, to some capacity. However, we found that the bags of words representation yielded a higher accuracy than either 2-grams or 3-grams. Additionally, we ultimately decided not to focus on POS frequency as it only proved useful in the polar-neutral classification (and less useful in the overall 3-way classification).

# 5 Classifier Results

| Classifier | Train Acc | Test Acc |
|---|---|---|
| textblob | — | 0.33 |
| VADER | — | 0.54 |
| SVM all | 0.91 | 0.80 |
| SVM pos-neg | 0.97 | 0.93 |
| SVM pol-neut | 0.92 | 0.85 |

Figure 1: Different Classifier Training and Testing Accuracies

The table above presents training and testing accuracies for the various sentiment analysis tools we used on the data. As you can see, the existing sentiment analysis libraries available in Python (textblob and vaderSentiment) did not perform too well on our task. In fact, the textblob package did no better than random guessing. vaderSentiment fared a bit better, but still isn't particularly accurate. Our SVM performed admirably in comparison, with training and testing accuracies in the 80's to 90's range across all of the various classification tasks.

### 5.1 Error Distribution

Although we achieved decent performance with our classifier, we were curious what types of errors our classifier made most frequently. As a result, we looked into the distribution of the classifier's errors, which is summarized in Figure 2.

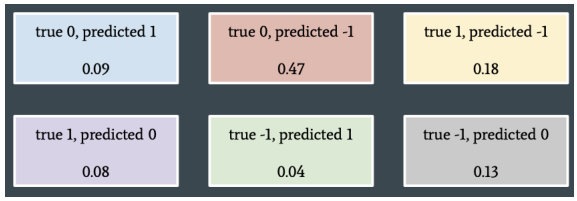


Figure 2: SVM all Classification Error Distribution

You'll notice our classifier makes almost half of all its errors on a single error case: when the tweet is ground truth neutral but it predicts the sentiment is negative. In fact, the classifier seems to overpredict tweets as negative in general, as the next most frequent error case is when the classifier predicts a tweet is negative when it is actually positive. While we were originally puzzled by this behavior, taking a closer look at the distribution of the data set provided some explanations. We found that the data is 16% positive tweets, 21% neutral tweets, and 63% negative tweets. This means we have 3 times the negative tweets as we do either positive or neutral, leading our classifier to overpredict negatively.

### 5.2 Error Confidence

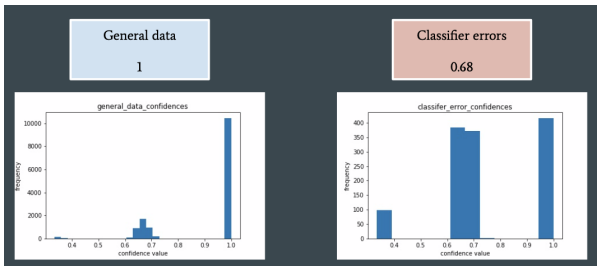


Figure 3: Set Diagram of User Sentiment

In addition to the types of errors our classifier made, we were also curious about whether our classifier struggled with the same tweets that humans struggled with. As a result, we examined the confidence values of the data points our SVM misclassified and compared them to the confidence values of the data set as a whole. We found that the median confidence value of the whole data set is 1.00, meaning the crowd unanimously agreed on the label of most of the tweets. However, the median confidence value of the classifier errors ended up being 0.68, implying our classifier struggled with the same tweets that humans also struggled with (these findings are summarized in Figure 3). While this is a promising sign for the performance of our algorithm, this highlights a key issue: how can we expect our classifier to correctly classify tweets that even a group of humans struggled to classify?

## 6 Practical Challenges

### 6.1 Effect of Humans on Data Collection

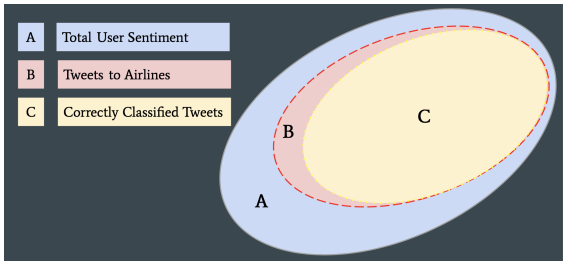


Figure 4: Set Diagram of User Sentiment

Humans-in-the-loop affected our project's data collection in two distinct ways: 1) the source of our data is inherently crowdsourced (being a series of tweets scraped from Twitter) and 2) the sentiment labels were inferred by another crowd of people. We see there are two layers of abstraction between the data we collected and the actual total sentiment of the

crowd. Think of the layers as filters – "I'm only going to grab the sentiments of people that tweet at airlines. Then, of those sentiments I'll only be able to use sentiments that could be confidently inferred by the crowd based on the text of the tweet." This relationship is shown by the diagram in Figure 4, where the blue set A contains the true distribution of all user sentiments. We'd like to sample this distribution directly. However, we only have access to the red set B, which is a distribution of the sentiments of the people that tweet at airlines. We've introduced sampling bias here by only selecting the subset of sentiments of people that decide to tweet at airlines, which skews heavily negative. Additionally, the only data that's useful to our classifier are the correctly inferred sentiments, which is an even smaller set represented by yellow set C. The second problem of mislabelling tweets isn't too bad of a problem, so expanding yellow set C to completely fill out B is mostly a label aggregation problem. However, the first problem of expanding set B to cover set A is much more intractable.

The intractable element of the problem stems from very natural human behavior: it makes sense that people are more likely to tweet an airline about a negative experience than a positive one, and it's also more likely that people are tweeting a logistical question than praising a flight. Because the data we collected wasn't balanced, the predictions of the algorithm were skewed negative. These are the kinds of things you *need* to think about when collecting data. If you don't, even without meaning to, you can end up creating a biased algorithm. While this may not seem like a huge problem in the context of analyzing the sentiment of airline tweets, it can become a huge problem if you're a bank trying to create an algorithm that does not discriminate on the basis for race when assigning loans.

## 6.2 Potential Remedies

We are going to outline two paths one could investigate to attempt to fix this problem. The two options are to throw out data from the over-represented class or generate data for the underrepresented class.

### 6.2.1 Throw Out Over-represented Data

Although a generally bad practice, one way to overcome the skew of the collected data is to even out the proportions of the data by tossing out some data points from the over-represented class. If we have proportions of $0.16, 0.21, 0.63$ for positive, neutral, and negative respectively, we can just toss out $\frac{2}{3}$ of the negative tweets to balance it out. Although this tends to lead to worse performance overall, it actually didn't affect the testing accuracy in our case (staying at around 0.80). However, running this chopped classifier on the whole data set leads to an unsurprising issue: the chopped classifier now has a tendency to overpredict things as neutral when they are actually negative, as shown in Figure 5. So, unfortunately, this didn't help us fix our problem.

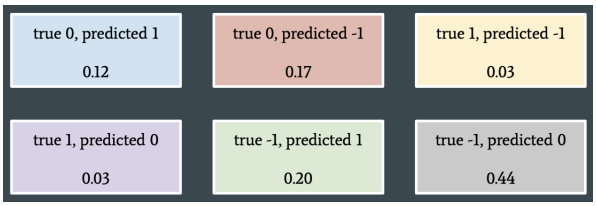


Figure 5: Set Diagram of User Sentiment

### 6.2.2 Generate Extra Underrepresented Data

On the other side of the coin, we can instead try to generate more positive and neutral data to make the proportions match. One possible way to do this is with the use of Generative Adversarial Networks (GANs), which generate extra fake examples based on what's available in the data set. Although

we did not attempt to do this due to time constraints, note that this method has a tendency to introduce extra noise into the data set, possibly lowering the classification accuracy. GANs could also skew your classification, because the fake examples will all be based on the small subset of original examples from the underrepresented class.

### 6.3 Future Work

The two remedies represent different sides of the same data coin: we can try raising the underrepresented classes up or bring the overrepresented classes down. If we had more time to pursue this project, we would be interested in actually using GANs to see how they might fare. We would also be interested in how the use of various label aggregation methods and additional polling for low confidence tweets might affect the results.

## 7 Conclusion

This project attempts to communicate some general problems one may run into using crowdsourcing in supervised machine learning, learned from our own experiences creating a sentiment analyzer for Twitter data.

If we had to choose one main takeaway from this project, it would be to approach the data collection process with caution and a problem-solving mindset. Two huge points to consider for a person collecting and working with corwdsourced data are:

1. Designing data collection methods that capture data points as close to the ground truth distribution as possible, so as to not introduce bias.

2. Designing good label aggregation methods to get the most efficient use of the data collected, so all of it can be effective in training your algorithm.

Keeping these two things in mind is essential to the art of working with data, and we hope their importance was highlighted through this specific example.

## References


[1] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[2] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, 2010.

[3] Twitter u.s. airline sentiment. Crowdflower, 2015.