Large Scale Sentiment Learning with Limited Labels

Vasileios Iosifidis
Leibniz University Hannover
& L3S Research Center
iosifidis@l3s.de

Eirini Ntoutsi
Leibniz University Hannover
& L3S Research Center
ntoutsi@l3s.de

ABSTRACT

Sentiment analysis is an important task in order to gain insights over the huge amounts of opinions that are generated in the social media on a daily basis. Although there is a lot of work on sentiment analysis, there are no many datasets available which one can use for developing new methods and for evaluation. To the best of our knowledge, the largest dataset for sentiment analysis is TSentiment [8], a 1.6 millions machine-annotated tweets dataset covering a period of about 3 months in 2009. This dataset however is too short and therefore insufficient to study heterogeneous, fast evolving streams. Therefore, we annotated the Twitter dataset of 2015 (228 million tweets without retweets and 275 million with retweets) and we make it publicly available for research. For the annotation we leverage the power of unlabeled data, together with labeled data using semi-supervised learning and in particular, Self-Learning and Co-Training. Our main contribution is the provision of the *TSentiment15* dataset together with insights from the analysis, which includes a batch and a stream-processing of the data. In the former, all labeled and unlabeled data are available to the algorithms from the beginning, whereas in the later, they are revealed gradually based on their arrival time in the stream.

CCS CONCEPTS

• **Information systems → Sentiment analysis**; • **Computing methodologies → Semi-supervised learning settings**;

KEYWORDS

semi-supervised learning, sentiment analysis, self-learning, co-training

ACM Reference format:
Vasileios Iosifidis and Eirini Ntoutsi. 2017. Large Scale Sentiment Learning with Limited Labels. In *Proceedings of KDD '17, Halifax, NS, Canada, August 13-17, 2017,* 10 pages.
https://doi.org/10.1145/3097983.3098159

1 INTRODUCTION

A huge amount of opinions is generated on a daily basis referring to essentially every subject - to products, persons, brands, events and topics. Opinions are valuable for consumers, who benefit from the experiences of other consumers, in order to make better buying decisions but also for vendors, who can get insights on what customers like and dislike [12]. Such sort of data are freely available nowadays, however due to their amount and complexity a proper analysis is required in order to gain insights.

Sentiment analysis aims at characterizing the sentiment content of a text as either positive or negative (some approaches also consider the neutral class). Most of the approaches in this area work with supervised learning, thus presuppose full availability of labeled instances. In reality though, it is difficult, expensive and time-consuming to obtain labels for all these instances, as they require experienced human annotators.

Semi-supervised learning addresses this problem by leveraging unlabeled data, together with the labeled data, to learn classification models. In this work, we employ two well known semi-supervised learning approaches, Self-Learning and Co-Training, in order to annotate a huge collection of tweets.

Our contributions are summarized below:

- We create *TSentiment15*[1], a collection of 228 Million tweets without retweets and 275 Million tweets with retweets, collected from Twitter using its public streaming API which spans the whole year 2015.
- We extensively evaluate the performance of Self-Learning and Co-Training and how it is affected by the amount of labeled data, the amount of unlabeled data and the confidence threshold.
- We process the data in two different ways: as a batch, where both labeled and unlabeled data are available to the algorithm from the beginning and as a stream, where both labeled and unlabeled data are gradually available to the algorithms as they arrive from the stream. We show that streaming achieves a comparable to the batch approach accuracy, while being more efficient.

The rest of the paper is organized as follows: Related work is presented in Section 2. In Section 3, we describe our dataset and how we derived our ground truth for learning. The semi-supervised learning approaches are described in Section 4. Our experiments are described in Section 5, conclusions and outlook in Section 6.

2 RELATED WORK

Our related work comes from different areas: semi-supervised learning, large scale annotations and sentiment analysis.

Nigam et al. [16] propose an algorithm for learning from labeled and unlabeled documents based on Expectation - Maximization(EM) and Multinomia Naive Bayes(MNB). The algorithm first trains a classifier using the available labeled data, and probabilistically labels the unlabeled ones. It then trains a new classifier using the labels of all the documents, and iterates until convergence. This basic algorithm was improved by two extensions: a weighting factor to

[1] Available at: https://l3s.de/%7Eiosifidis/TSentiment15/

modulate the contribution of unlabeled data and by using multiple mixture components per class, instead of a single one. Su et al [19] propose a semi-supervised extension of MNB, SFE, that uses the estimates of word probabilities obtained from unlabeled data and class conditional word probabilities learned from the labeled data, to learn the MNB parameters. Lucas et al. [13] introduced MNB-FM a method that extends MNB to leverage marginal probability statistics of each word, computed over the unlabeled data. The marginal statistics are used as constraints to improve the class conditional probability estimates for the positive and negative classes. Zhao et al. [25] proposed MNB-WSC, which preserves reliable word estimations, as extracted from a fair amount of labeled data. We use MNBs as our based model. Despite their unrealistic conditional independence assumption, they are known to perform moderately and in some cases, it has been reported that their performance for short texts is equal or superior to other more complex models [23].

A comprehensive survey of semi-supervised learning approaches for Twitter is provided in this recent survey [18]. Similarly to us, they found that Co-Training performed better with limited labels, whereas Self-Training is the best choice when a significant amount of labeled tweets is available. In contrast to the existing small scale datasets they used for evaluation, we report on a huge collection covering the whole year 2015. Dasgupta et al. [5] experimented with a large variety of algorithms for semi-supervised sentiment classification, including active learning, spectral clustering and ensemble learning. Cross-domain sentiment classification is proposed in [1], where the authors exploit a small number of labeled and a huge amount of unlabeled data using EM.

Closely related to semi-supervised learning is self-learning [6]. The central idea behind self-learning is that we can expand the training set by using the confident predictions of the classifier. Self-learning might cause error propagation as the predictions of the classifier are then used for its training [10, 26]. To deal with these issues Co-Training was introduced in [4] which combines different views of the data in two classifiers, which are then used for the expansion of the training set. The intuition behind this approach is that different classifiers will make different errors and therefore one classifier can learn from the other instead of just learning by itself as in self-learning. In [11] the authors study class imbalance for semi-supervised classification. They use under-sampling in order to generate multiple balanced training sets and during the iterations of the semi-supervised process they dynamically change the classifiers by varying the feature space. In a recent work [24], the authors use the negations and antonyms to generate an opposite view of the training data; the original and the opposite view are exploited afterwards via Co-Training.

TSentiment [8] is a dataset of 1.6 million tweets from the time period between April 6, 2009 to June 25, 2009, which are annotated as positive and negative through distant supervision. The training set consists of tweets with emoticons, which serve as noisy labels. To the best of our knowledge this is the largest Twitter dataset for sentiment analysis and is used extensively also in stream mining due its temporal aspects [3, 22]. HSpam14 [17] is a dataset of 14 million tweets in English which are annotated with spam and ham (or, non-spam) labels. The annotation process consists of four steps: a heuristic-based selection of tweets that are more likely to be spam, a cluster-based manual annotation, a reliable ham tweet detection and finally, EM-based label prediction for the rest of the unlabeled tweets.

3 DATASET DESCRIPTION

Our dataset, the preprocessing we applied and its effect on the dataset are described in Section 3.1. In Section 3.2 we describe how we derive the training set, i.e., labeled instances for the classification task. In Section 3.3 we describe the distribution of labeled and unlabeled data over the course of the monitoring period.

3.1 Data collection and preprocessing

We collected data from Twitter using its public streaming API[2], which provides a random selection of tweets (about 1% of all tweets); our monitoring period is the whole 2015.

In total, 1.9 billion tweets were crawled, in all different languages (English, Japanese, Spanish, Greek, etc.). We selected only the English tweets which were not re-tweets (as flagged by the API); the filtered dataset consists of 384 millions tweets (20%), which generate 269 million distinct words.

We applied several preprocessing steps whose effect is depicted in Figure 1:

- *Slang words replacement*: We mapped slang words to normal expressions using a slang word dictionary[3]. For example, "lol" was mapped into "laughing out loud". This resulted in a slight increase of the words.
- *Links and mentions*: Links and mentions, e.g., "https://example.com", "@bbc", were removed.
- *Negation handling*: We consider negations on verbs and adjectives. For the former, we concatenated to a single verb, e.g., "don't work"' → 'not_work". For the later, we replaced the negation with its antonym, e.g., "not bad"→ "good", using available lists.
- *Special character removal*: We removed punctuation and numbers. We removed the symbol '#' from hashtags and we treated them as normal words. We replaced repeated letters occurring more than two times in a row with two letters; e.g., "huuuungry" → "huungry".
- *Removal of emoticons*: We also removed the emoticons from training data, aiming at classifiers that can learn from the word features. In general, we removed all non ascii characters most of which were special types of emoticons. But, we use the emoticons to derive the labels for the training set (c.f., Section 3.2).
- *Stopword removal*: We removed stopwords using Weka's stopwords list [4].
- *Stemming*: We applied Porter stemmer.
- *Removal of rare words*: We removed rare words from the corpus, using a frequency of 10 as the cut-off value.
- *Removal of short tweets*: Finally, we removed those tweets with less than four (<4) words after preprocessing, similarly to [20].

After preprocessing the tweets are represented on average with less words. In overall, the preprocessing resulted in a 41% reduction of the corpus (from 384M to 228M tweets). The distinct words were reduced by 99,5% (from 269M to 1,17M distinct words).

[2]https://dev.twitter.com/streaming/overview
[3]www.noslang.com
[4]http://weka.sourceforge.net/doc.dev/weka/core/stopwords/Rainbow.html

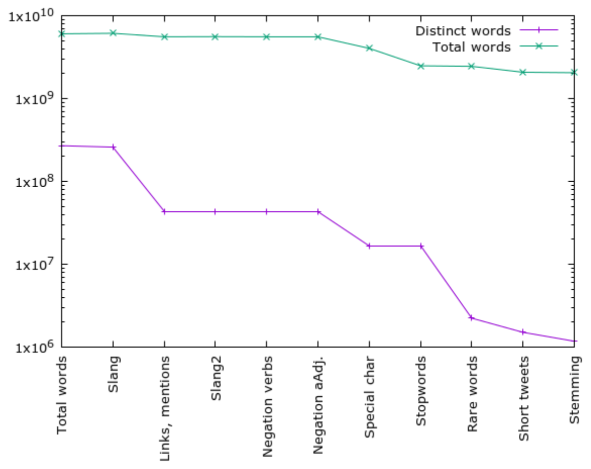

Figure 1: Preprocessing effects

3.2 Ground truth for learning

In the absense of human labels, we derive our ground truth for learning by considering two options: i) an emoticon-based approach and a sentiment lexicon approach (using SentiWordNet[5]). Our idea is to consider as groundtruth those tweets for which both emoticons and SentiWordNet agree in their labelings.

Deriving labels from emoticons. Using a list of positive[6] and negative[7] emoticons we identify tweets with clear sentiment; those are tweets with only positive or only negative emoticons, which we then classify accordingly. This approach is similar to [9].

In our 220M tweets dataset, only 10,1M (4.4%) contain emoticons. Out of 10.1M tweets with emoticons, 3,8M (37%) were classified as clear positive (those with only positive emoticons), 1,5M (15%) as clear negative (those with only negative emoticons), 4,8M (48%) as mixed cases (both positive and negative emoticons). Only tweets with clear emoticon-based sentiment, a total of 5.3 million tweets, were used for building the ground truth.

Deriving labels from SentiWordNet. SentiWordNet [2] is a dictionary of words and their associated sentiment. The words might appear multiple times as different part of the speech (POS). Therefore we first find the POS of each word in a tweet using Stanford's POS tagger [21] using the whole tweet to derive the context. Then, we calculate for each tweet its overall score by aggregating the scores of its component words from SentiWordNet; for the aggregation, each word is weighted with a harmonic series [7].

Building the ground truth. The results of juxtaposing the emoticons- and SentiWordNet- based labels are displayed in the confusion matrix in Table 1. We considered as our ground truth the true positives, i.e., tweets where emoticon-based and SentiWordNet-based labeling agree. Our final ground truth dataset consists of 2,527,753 tweets.

[5]http://sentiwordnet.isti.cnr.it/

[6]Positive emoticons : c) =] :] :} ; > :>) :^) : D =) ;) :) 8) (: (; : o) : −) : P < 3 : 3 ^_^

[7]Negative emoticons : −c : [: { :< : −(: / : −[: c : − < : (:′ { >: [

From these 2,5M tweets, 87.47% are positives (2,211,091 tweets) and the rest 12.52% (316,662 tweets) are negatives.

Table 1: Confusion Matrix: Emoticon-vs SentiWordNet-based labels

	SWN. Pos.	SWN. Neg.	SentiW. Neutral
Emot. Pos.	2,211,091	840,787	807,887
Emot. Neg.	1,032,536	316,662	157,322

3.3 Temporal distribution of the dataset and the labels

The temporal distribution of our dataset is depicted in Figure 2 including both labeled tweets (according to Section 3.2) and unlabeled ones.

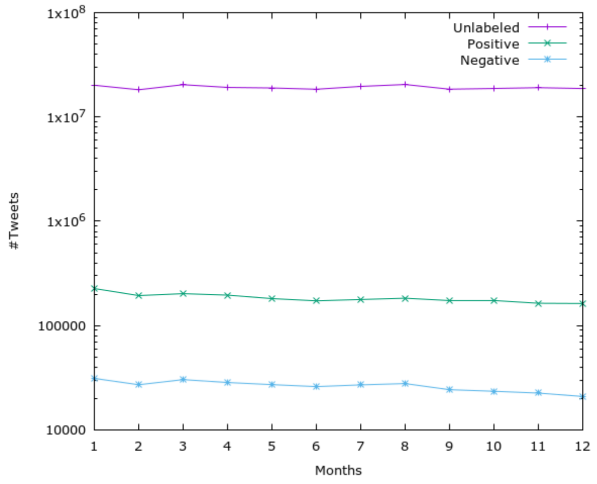

Figure 2: Dataset distribution on a monthly basis

As we can see, the amount of unlabeled tweets is vast compared to the amount of labeled tweets and this holds across the timeline. On monthly average, the unlabeled set is 82 times larger than the labeled set. For the labeled tweets, the negative class is miss-represented comparing to the positive class; the average ratio of positive to negative tweets per month is 3. Finally, there are no gaps in the monitoring period, i.e., we have both labeled and unlabeled tweets for each month.

4 LEARNING WITH LIMITED LABELS

In our application, we have a small number of labeled instances, L, and a huge number of unlabeled instances U; this is a typical scenario in many real life applications as despite the huge amounts of data nowadays, only a small fraction of this data is labeled and therefore can be directly used for (supervised) learning.

To deal with this issue, several approaches exist which except for the labeled data also leverage unlabeled data. In this work, we investigate two popular semi-supervised learning approaches: Self-Learning and Co-Training, described hereafter.

4.1 Self-learning

The main idea in Self-Learning [6] is to use the labeled set L to build a classifier, then iteratively apply the model to the unlabeled corpus U and in each iteration, expand the training set with instances from the unlabeled corpus which were predicted with high confidence, according to a threshold δ, by the classifier. The pseudocode of the Self-Learning algorithm is shown in Algorithm 1.

The initial training set is the labeled set L. The training set is expanded in each iteration, by including confident predictions from U and is used for building a new classifier. The procedure continues until U is empty or after a certain number of iterations or, if no further expansion is possible.

```
**Input:** $L$: labeled set, $U$: unlabeled set, $\delta$: confidence threshold
$T \longleftarrow L$
while (stopping criterion) do
    $\Phi \longleftarrow$ train classifier on $T$;
    for i=1 to $|U|$ do
        if (confidence of $\Phi$.classify($U_i$) $\geq \delta$) then
            $T \longleftarrow T \cup U_i$
            Mark $U_i$ as labeled
        end
    end
    update $U$ by removing labeled instances;
end
return $T$;
```

Algorithm 1: Pseudocode of Self-Learning

The intuition behind Self-Learning is that we can use the confident decisions of the classifier to expand the training set, in some sort of exploitation of what the classifier already knows sufficiently well. However, since some of the these predictions might be erroneous, Self-Learning might cause error propagation as at the end the training set is a mix of original labels and predictions which are taken equally into account for learning [10, 26]. Moreover, since the classifier mainly exploits what it already knows and expands the training set through similar instances, it is more difficult to learn new concepts.

4.2 Co-Training

Co-Training was introduced in [4] and it assumes that the feature space, lets denote it by X, can be split into two parts, $X = (X^1, X^2)$, called "views". The Co-Training algorithm trains two classifiers Φ^1, Φ^2, each working exclusively on one view, X^1, X^2, respectively. Initially, both classifiers are trained over the labeled set L, but each on its own view, X_i, $i \in \{1, 2\}$. The unlabeled data are utilized as follows: At each iteration, Φ^1 classifies a few unlabeled instances for which he is more confident about and appends them to the training set. Similarly for Φ^2. Co-Training then updates both classifiers with this additional "pseudo-labeled" data.

We follow a slightly different version, c.f., Algorithm 2. We initialize Co-Training as above, with Φ^1, Φ^2 classifiers built upon the labeled set L but each exclusively on one view, X^1, X^2, respectively. At each iteration of Co-Training, the most confident predictions of each classifier are used for the expansion of the training set of the other classifier. That is, the most confident predictions of Φ^1 are used to expand the training set of Φ^2 and vice versa.

Therefore, although both classifiers start with the same training set L, over the iterations and as one learns from the other, the training sets of the two classifiers are different. The procedure stops if U is empty or after a certain number of iterations or, if no further expansion is possible.

```
**Input:** $L$: labeled set, $X^1, X^2$: the two feature views, $U$:
        unlabeled set, $\delta$: confidence threshold
$T_1, T_2 \longleftarrow L$
while (stopping criterion) do
    $\Phi_1 \longleftarrow$ train classifier on $T_1$ (using $X_1$ view);
    $\Phi_2 \longleftarrow$ train classifier on $T_2$ (using $X_2$ view);
    $TempSet_1 = \emptyset$;
    $TempSet_2 = \emptyset$;
    for i=1 to $|U|$ do
        if (confidence of $\Phi_1$.classify($U_i$) $\geq \delta$) then
            $TempSet_1 \longleftarrow TempSet_1 \cup U_i$
            Mark $U_i$ as labeled
        end
        if (confidence of $\Phi_2$.classify($U_i$) $\geq \delta$) then
            $TempSet_2 \longleftarrow TempSet_2 \cup U_i$
            Mark $U_i$ as labeled
        end
    end
    update $U$ by removing labeled instances;
    $T_1 \longleftarrow T_1 \cup TempSet_2$;
    $T_2 \longleftarrow T_2 \cup TempSet_1$;
end
```

Algorithm 2: Pseudocode of Co-Training

The intuition behind Co-Training is that each classifier provides labeled data to the other classifier, which the later can use for learning. In contrast to Self-Learning, in Co-Training a classifier does not learn by its predictions rather by the confident predictions of the other learner (the first classifier might be non-confident for those predictions). Thus the two views (classifiers) working together manage to progress learning. The assumption is that each view (classifier) is able to learn the target concept given enough data, that is, each view is sufficient for learning on its own.

5 EXPERIMENTS

5.1 Experimental settings

We conduct the experiments on the Twitter dataset we introduced in Section 3. As already mentioned, our base classifier is Multinomial Naive Bayes (MNBs). For the implementation we used Spark's distributed environment and its machine learning library MLlib [14].

Features for Self-Learning and Co-Training. For the Self-Learning approach, we used unigrams (after pre-processing, as described in Section 3).

For Co-Training, we need two different feature sets. Therefore, except for the unigrams we also experimented with two different setups: (i) bigrams (the feature space was extracted from the pre-processed text, as described in Section 3), (ii) language features: we extracted part-of-speech tags using [21]. Furthermore, we counted words in capital, words with repeated characters, links and mentions. We also employed a dictionary which contained sentimental hashtags [15] and counted the occurrences of positive and negative hashtags in our tweets, if any (the extraction was done over the original text of tweets, not the preprocessed ones). We refer to this feature space as "SpecialF".

Therefore we have two alternative Co-Training setups:

- Co-Training$^1_{\text{[unigrams-bigrams]}}$
- Co-Training$^2_{\text{[unigrams-SpecialF]}}$

5.2 Performance of batch annotation

5.2.1 Experimental settings. We split our ground through (2.5 million tweets) in 10 folds, 1 of which is used for testing and the rest, together with unlabeled data predictions, are used for training. In each iteration of the 10-cross-validation process, the test set is fixed, the training set however is expanded through the addition of (unlabeled) tweets that were predicted with high confidence by the classifier. We report the averaged results of 10-fold cross-validation in terms of classification accuracy for both Self-Learning and Co-Training, under different confidence thresholds δ that determines which of the classifier predictions are incorporated in the training set.

5.2.2 Self-Learning-based batch annotation. In Figure 3 (top) we display the accuracy of the Self-Learning approach under different confidence thresholds δ, in the range [65%-100%] and how, the accuracy changes as the algorithm iterates through the remaining unlabeled tweets set. We stop at 5 iterations as the algorithm manages to annotate almost all unlabeled tweets within those iterations. We also show the accuracy in the initial training set, i.e., before expansion.

The accuracy of Self-Learning drops comparing to the accuracy of the initial model (trained in the initial labeled set); this is to be expected as the training set is expanded through predictions. The drop is more drastic in the first iteration. The reason is that the 1st iteration results in the largest expansion of the training set as a large number of predicted instances is added to the training set therefore affecting the extracted models. The expansion depends on the threshold δ, as higher values are more selective and therefore result in smaller expansion. The training set expansion under different δ thresholds and over the iterations is shown in Figure 3 (bottom). At δ=65%, for example, the expanded training set is about 8,100% larger than the original training set L.

The accuracy drops with δ. The decrease is directly related to the amount of expansion of the training set. For the low δ values, the decrease is very small after the first two iterations; the reason is that the bulk of predictions was already added to the training set in the first two iterations and therefore the new predictions additions do not really influence the classifiers. For larger δ values (90%-95%) though, the accuracy drops faster as the corresponding training set expands more gradually. The only exception is $\delta = 100\%$; the accuracy does not change because the training set is hardly influenced, as this threshold is very selective and therefore not many predictions can satisfy it.

The annotated dataset is depicted in Table 2: for different δ we report the amount of positive, negative and unlabeled tweets, i.e., tweets that remained unlabeled after the fifth iteration. As we can

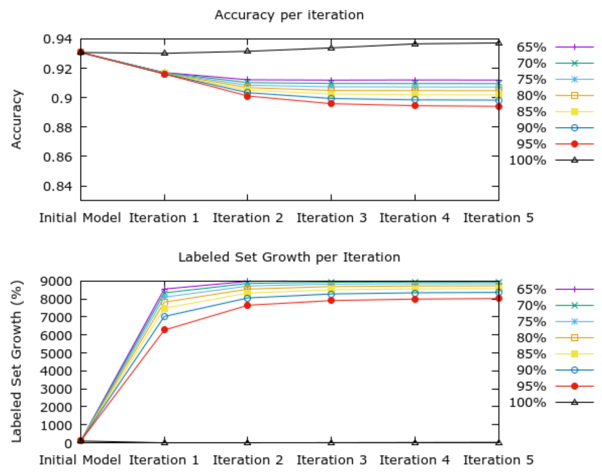

Figure 3: Batch annotation with Self-Learning: accuracy and labeled set growth under different δ values while the algorithm iterates through the remaining unlabeled set.

Table 2: Batch annotations with Self-Learning: Annotated results per class for different confidence values δ.

δ	positive predictions	negative predictions	unlabeled
65%	201,860,127 (88.46%)	26,315,605 (11.53%)	1.13%
70%	200,212,418 (88.49%)	26,033,446 (11.50%)	1.97%
75%	198,296,101 (88.59%)	25,525,791 (11.40%)	3.02%
80%	196,017,401 (88.78%)	24,757,934 (11.21%)	4.34%
85%	193,134,363 (89.06%)	23,720,362 (10.93%)	6.03%
90%	189,271,805 (89.49%)	22,217,878 (10.50%)	8.36%
95%	183,012,328 (90.21%)	19,843,802 (9.78%)	12.10%
100%	650,450 (99.86%)	877 (0.13%)	99.71%
Initial Model	2.211.091 (87,47%)	316.662(12,52%)	

see the more selective δ is the more tweets remain unlabeled, with the extreme case of $\delta = 100\%$ where almost all tweets (99.71%)remained unlabeled.

We report the amount of positive and negative annotations over the labeled set and not over the complete dataset, in order to highlight the class distribution of the predicted labels. In the last row of the table we also report the class distribution in the original training set, i.e., before expansion. The majority of the predictions refers to the positive class, on average 88% of the predictions are positive and 11% negative. As the confidence threshold increases, the positive class percentage in the predictions also increases. The higher percentage of positive class predictions (99,86%) is manifested with a threshold of 100%, implying that the classifier is more confident about the positive class and therefore the training set is augmented with more examples of the positive class.

5.2.3 Co-Training-based batch annotation. Figure 4 demonstrates the accuracy of Co-Training for two confidence levels ($\delta = 65\%$ and $\delta = 95\%$) and how the accuracy changes as the algorithm iterates through the remaining unlabeled tweets set. We stopped at four iterations since after the 3rd iteration the number of unlabeled tweets that are labeled given the specific δ is very small.

The best performance is achieved when we learn from unigrams (1st classifier) and bigrams (2nd classifier), i.e., by Co-Training$^1_{\text{[unigrams-bigrams]}}$. Hereafter we use this classifier for the comparison and we refer to it

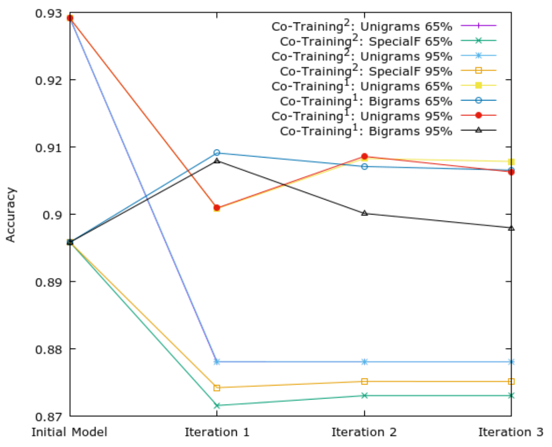

Figure 4: Batch annotation with Co-Training: accuracy for δ = 65%, δ = 95% while the algorithm iterates through the remaining unlabeled set. The accuracy is displayed for each Co-Training classifier-member.

Table 3: Average accuracy of Co-Training$^1_{[\text{unigrams-bigrams}]}$ members under different δ.

δ	Unigrams	Bigrams
65%	90.65%	90.71%
70%	90.76%	90.66%
75%	90.81%	90.57%
80%	90.82%	90.51%
85%	90.78%	90.41%
90%	90.69%	90.28%
95%	90.50%	90.02%
100%	93.16%	89.03%
Initial Model	93.07%	88.52%

as Co-Training. We show the accuracy of this model under different thresholds δ in Table 3. We also show the accuracy of the initial model, i.e., before the training set expansion; the expansion results in accuracy loss (around 2%). As we increase δ, there is a small improvement for values in the range 65%-80%, but the performance slightly drops with higher values in the range 85%-95%. The only exception (which outperforms the initial model) is δ = 100% which is not affected that much as the training set is not much expanded due to the very selective δ. Moreover, as we can see in Table 4, such an expansion refers mainly to instances of the majority class (positive).

How the performance varies over the different iterations and for different thresholds δ and what degree of original training set expansion is achieved is shown in Figure 5 (top). The picture is similar to Self-Learning, the first two iterations produce the largest amount of confident predictions, especially for lower δ values.

The annotated dataset is depicted in Table 4. Similarly to what we observed for Self-Learning, the more selective δ is the more tweets remain unlabeled, at δ = 100% almost all tweets (99.44%) remained unlabeled. Moreover, the amount of unlabeled tweets is smaller comparing to the Self-Learning case 2. Regarding the class distribution of the predictions, the positive class is still predicted more often. However and on the contrary to Self-Learning, the negative class is better represented in this setting. The explanation

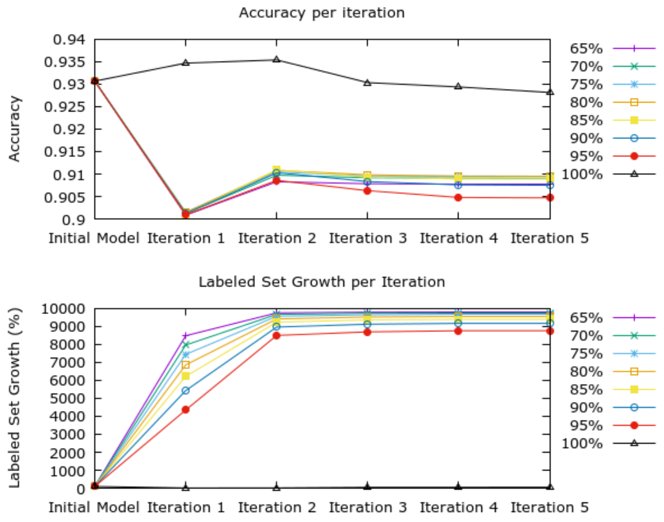

Figure 5: Batch annotation with Co-Training: accuracy and labeled set growth under different δ values while the algorithm iterates through the remaining unlabeled set.

Table 4: Batch annotation with Co-Training: Annotated results per class for different confidence values δ.

δ	positive predictions	negative predictions	unlabeled
65%	175,704,567 (76.64%)	53,547,361 (23.35%)	0.66%
70%	178,361,861 (78.26%)	49,544,295 (21.73%)	1.25%
75%	180,646,395 (79.90%)	45,419,649 (20.09%)	2.04%
80%	182,180,488 (81.52%)	41,287,186 (18.47%)	3.17%
85%	182,758,504 (83.04%)	37,300,375 (16.95%)	4.65%
90%	182,707,849 (85.06%)	32,069,200 (14.93%)	6.93%
95%	179,527,239 (87.43%)	25,810,993 (12.56%)	11.02%
100%	1,281,748 (99.60%)	5,116 (0.39%)	99.44%
Initial Model	2.211.091 (87,47%)	316.662(12,52%)	

Table 5: Batch: Self-Learning vs Co-Training, average accuracy for different δ

δ	Self-Learning	Co-Training (Unigrams)
65%	91.30%	90.65%
70%	91.11%	90.76%
75%	90.93%	90.81%
80%	90.75%	90.82%
85%	90.55%	90.78%
90%	90.31%	90.69%
95%	90.03%	90.50%
100%	93.38%	93.16%
Initial Model	93.07%	93.07%

lies on the fact that in Co-Training classifiers learn from each other, rather than only from their predictions.

5.2.4 *Self-Learning vs Co-Training.*

In Table 5 we report the average (over all iterations) accuracy of Co-Training and Self-Learning for different δ values (for the Co-Training, we report here on the performance of the best classifier member, i.e., the one built upon unigrams).

As we can see, Self-Learning accuracy decreases with δ much faster than the accuracy of Co-Training; both achieve improvements at 100%. As we can see from Tables 2, 4, Co-Training produces more labels than Self-Learning and better class balance.

Thus far we expand the training set based on the confidence threshold δ, which however results on an uncontrolled expansion (in terms of number of instances) of the training set. To evaluate the effect of the magnitude of dataset expansion, we performed a controlled experiment where we gradually expand the training set by adding classifier predictions. In particular, we built an initial model in the original training set which we then used to annotate the unlabeled set. From this set we then randomly selected [10%-100%] predictions/instances which we used for dataset expansion. The results are depicted in Figure 6. As we can see the accuracy drops as we further expand the dataset, for all methods. As the ratio of predicted to labeled instances increases, Co-Training (bigram's model) experiences a faster drop in its performance.

We also evaluated the effect of labeled data, by varying the amount of labeled data and using a 10% sample of the unlabeled set for predictions (which scores the best performance in Figure 6). The results are depicted in Figure 7. As we can see, when the number of labels is small Co-Training performs better than Self-Learning. With 40% of labels or more Self-Learning is marginally better.

5.3 Performance of stream annotation

5.3.1 Experimental settings. For the stream approach we process the data on a monthly basis and we evaluate how the temporal processing affects our methods. Let L_i be the labeled data (ground truth) for month i and let U_i be the corresponding unlabeled set. Our complete dataset therefore is a sequence of the form: $((L_1, U_1), (L_2, U_2), \cdots (L_{12}, U_{12}))$ covering the whole year 2015.

We evaluate two variants:

- *without history*: we learn our models (Self-Learning, Co-Training) on each month i based on the labeled data of that month L_i and also by including confident model predictions from the corresponding unlabeled dataset U_i. We evaluate those models with the ground truth for the next month L_{i+1}.
- *with history*: for a month i, the labeled set upon which we build our model consists of all labeled instances up to month i, i.e., $\sum_{i=1}^{i} L_i$. Similarly, for the expansion we consider all unlabeled instances up to month i, i.e., $\sum_{i=1}^{i} U_i$ and we add to the training set those that were predicted with high confidence by the model.

That is, we differentiate on whether we use historical data from the stream to build our models or we just use data from the current timepoint (month).

In the above scenario all labeled data are used for training and testing, as each month is tested with the labeled data of the next month. We refer to this as *prequential evaluation*. We also consider a *holdout evaluation*: we split the original dataset into a training and a testing set. The evaluation procedure is similar to prequential evaluation, the only difference is that we use for training (testing) only data from the training (testing, accordingly) set. That is not all labeled data are used for training/testing, rather a sample of them according to the initial split.

5.3.2 Self-Learning vs Co-Training. The holdout evaluation is depicted in Figure 8, the prequential in Figure 9; the performance is similar for both evaluations. For both Co-Training and Self-Learning, history improves the performance. For the models with history, Co-Training is better in the beginning but as the history grows its performance decreases and Self-Learning results in best performance. So Co-Training is more effective with less labels; this is also evident in the non-history models, where we see that Co-Training outperforms Self-Learning for almost all months.

From the above experiment it is clear that history improves performance. To evaluate the effect of history's length, we run the same experiment with a sliding window of three months; for example, we used the labeled instances of months [1-3] for building an initial model, we expand the training set including predictions for unlabeled instances in months [1-3] and we use the derived model to score the next month, month 4. The results are depicted in Figure 10. As we can see, Self-Learning is better for almost all months. Again, we denote that Co-Training works better with limited labels.

Comparing to the full-history case, in the sliding window approach we have a small decrease in the performance (less than 2.0%) but on the other hand much more light models and therefore better efficiency (time, memory). The amount of data for each approach is depicted in Figure 11: labeled set is the original labeled data, training set is the expanded dataset of labeled instances and confident classifier predictions that was used for training.

As we can see, when we consider historical data, the amount of labeled and training instances is increasing over time, whereas for the non-history version these amounts are not changing that much over time. A similar behavior occurs for the sliding window version.

The class distribution of the predictions is shown in Figure 12, for all different window models: without history, with full history and with a sliding history of 3 months. For all window models, most of the predictions for both Co-Training and Self-Learning refer to the positive class. Co-Training produces on average less positive instances than Self-Learning. This is evident in the with-history and sliding-window approach: Self-Learning produces more positive predictions than Co-Training. This is due to the fact that Self-Learning is biased towards what it knows best (the positive class in this case). On the contrary, Co-Training is less biased as the two classifiers learn by each other.

To conclude the stream approach, Co-Training achieves the best performance with limited labels; as the amount of labeled data increases, Self-Learning surpasses its performance. Self-Learning is more biased to its own predictions comparing to Co-Training and therefore results in more positive predictions.

5.4 Performance with Duplicated Text (retweets)

Thus far, we reported on English tweets without redundancy (retweets). To evaluate the impact of redundancy on the aforementioned methods we repeat all our experiments with retweets. Due to lack of space, we report here on some of the results.

In Table 6, same setup as Table 5, we report on the performance of the redundant dataset. As we can see, the accuracy values are lower comparing to the non-retweets case. Moreover, the drop in the performance as δ increases is higher.

The effect of redundancy on the class imbalance of the predicted labels is shown in Table 7, where we display the negative:positive class ratio for different δ values, for Self-Learning and Co-Training

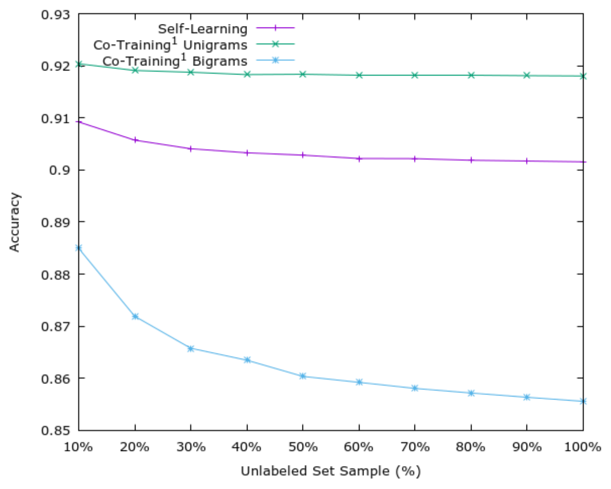

Figure 6: Batch: Effect of predicted instances used for dataset expansion

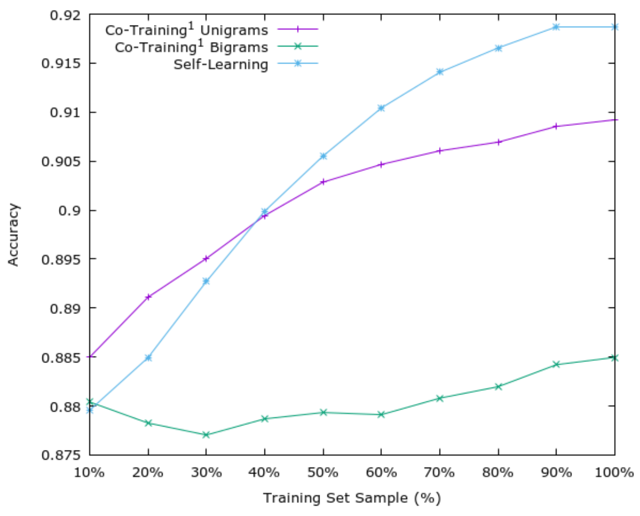

Figure 7: Batch: Effect of labeled set

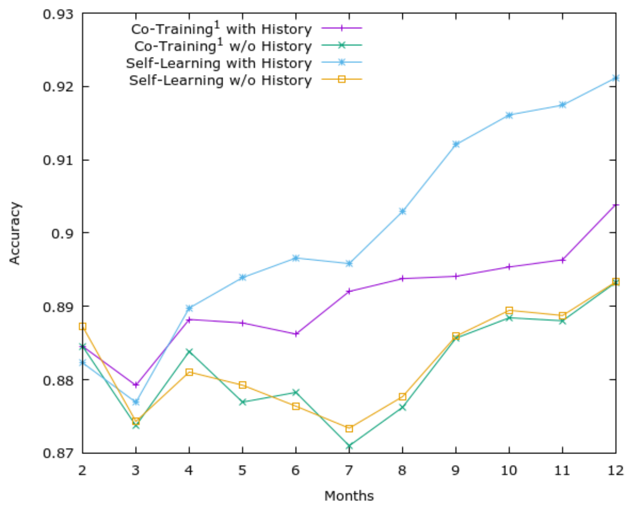

Figure 8: Stream: Holdout

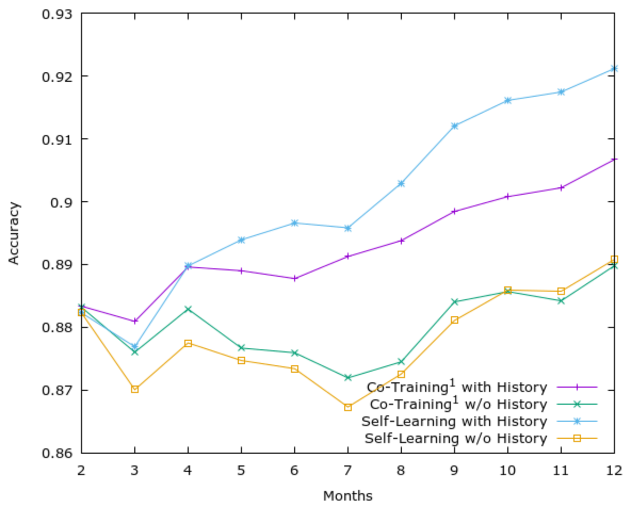

Figure 9: Stream: Prequential

Table 6: Batch: Self-Learning vs Co-Training, average accuracy for different δ using Retweets

δ	Self-Learning	Co-Training (Unigrams)
65%	87.09%	87.24%
70%	86.73%	87.09%
75%	86.42%	86.91%
80%	86.09%	86.68%
85%	85.75%	86.39%
90%	85.31%	86.04%
95%	84.71%	85.43%
100%	92.79%	91.25%
Initial Model	92.92%	92.92%

Table 7: Class Ratio (negative:positive) of the predictions for different datasets and methods

δ	Self-Learning noRts	Co-Training noRts	Self-Learning with Rts	Co-Training with Rts
65%	1:8	1:4	1:2	1:2
70%	1:8	1:4	1:2	1:3
75%	1:8	1:4	1:2	1:2
80%	1:8	1:4	1:2	1:2
85%	1:8	1:5	1:2	1:2
90%	1:9	1:6	1:2	1:2
95%	1:9	1:7	1:2	1:2
100%	1:741	1:248	1:2	1:14

for the dataset with retweets and the dataset without retweets. It is clear that the class imbalance is dramatically affected by duplicates. For both methods, the predictions are more balanced for the dataset with retweets. A possible explanation is that by eliminating the retweets, the negative class which was already a minority (75%-25% in the dataset with retweets) became even more underrepresented (87%-13% in the dataset without retweets). Since, we didn't explicitly handle the imbalance in the dataset, it manifested in the predictions of the MNB classifier.

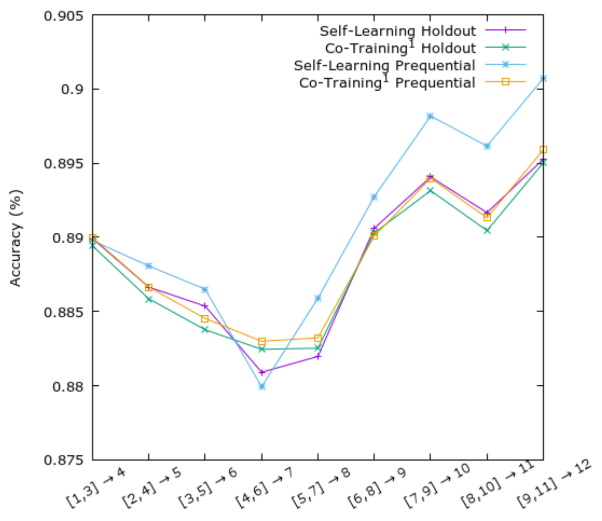

Figure 10: Stream: Sliding (3 months)

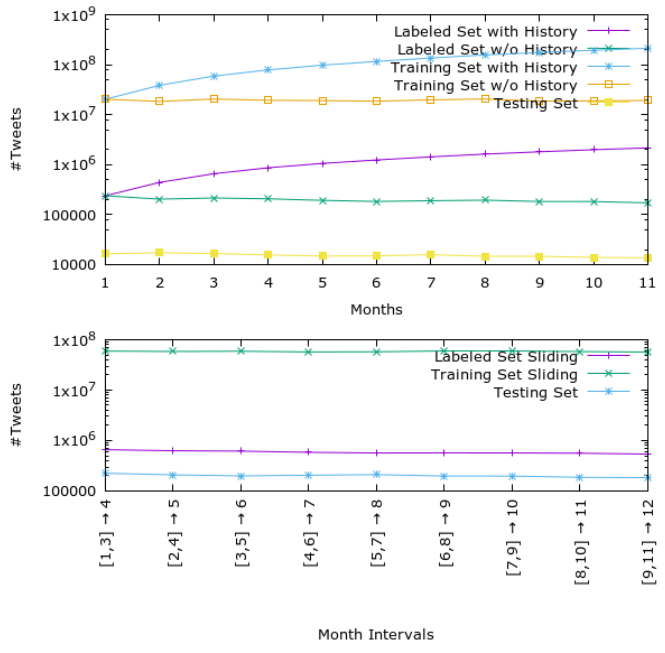

Figure 11: Stream: Prequential evaluation - cardinality of labeled, training, testing sets.

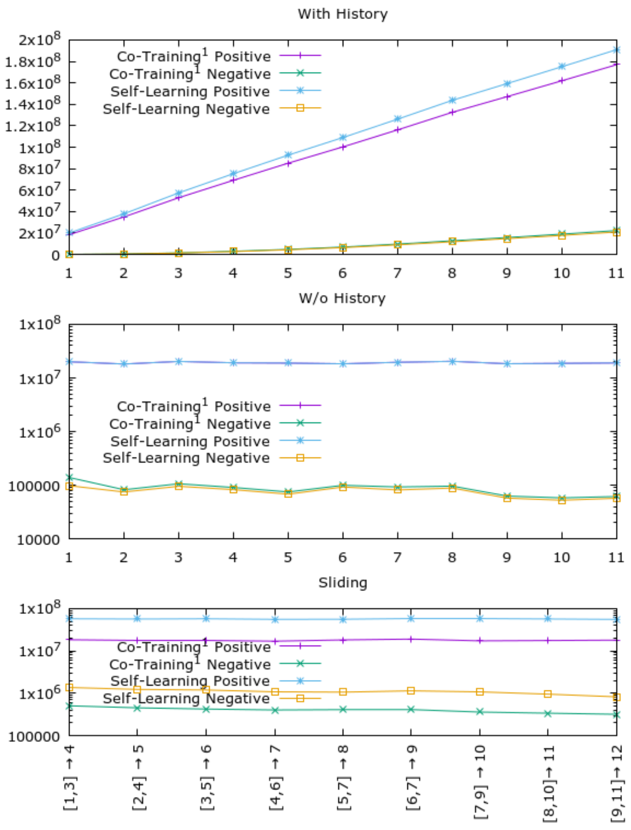

Figure 12: Stream: Class distribution of the annotated tweets over time

6 CONCLUSIONS AND FUTURE WORK

We presented how to annotate large scale collections with sentiment labels. Our case study is TSentiment15, 228 million tweets dataset with no retweets and 275 million tweets with retweets, for the whole year 2015 which we annotated using Self-Learning and Co-Training, using batch- and stream- processing. The motivation for this work is the lack of large scale labeled datasets that span large periods of time, especially important for stream mining research.

Despite the annotated datasets (with and without retweets) which we make available to the community, our analysis resulted in interesting insights:
Co-Training performs better than Self-Learning with limited labels. Although both Self-Learning and Co-Training benefit from more labeled data, after a certain point (40% labeled data, c.f., Figure 7) Self-Learning improves faster than Co-Training. Both approaches result in more positive predictions (c.f., Tables 2 and 4), thus favoring the majority class. Self-Learning moreover propagates the original class imbalance to the successive iterations (c.f., Table 2). This is not the case for Co-Training (c.f., Table 4). For streaming, (full) history helps with the performance. A sliding window-based history performs almost equally well, while employing less data, thus offering a good trade-off. The batch approach is better than streaming in terms of accuracy, however the later is much more efficient.

In this work, we didn't investigate the nature of the predictions; we plan to analyze the derived labels through crowd-sourcing to gain insights on the errors and success of the different methods. We will also investigate the disagreement between Emoticons- and SentiWordNet-based labelings (c.f., Table 1); SentiWordNet might be outdated, sentiment labels though might be noisy.

Moreover, we plan to investigate other ways of expanding the labeled set, like by constructing antonymous tweets of the opposite class from existing labeled tweets [24] or by grouping similar tweets into clusters and labeling the clusters [17]. We also plan to further investigate the issue of bias propagation in Self-Learning. As we saw, the original class imbalance is becoming more severe as the classifier learns by its predictions.

ACKNOWLEDGEMENTS

The work was partially funded by the European Commission for the ERC Advanced Grant ALEXANDRIA under grant No. 339233 and by the German Research Foundation (DFG) project OSCAR (Opinion Stream Classification with Ensembles and Active leaRners).

REFERENCES

[1] A. Aue and M. Gamon. Customizing sentiment classifiers to new domains: A case study. In *Proceedings of recent advances in natural language processing (RANLP)*, volume 1, pages 2–1, 2005.

[2] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.

[3] A. Bifet and E. Frank. Sentiment knowledge discovery in twitter streaming data. In *International Conference on Discovery Science*, pages 1–15. Springer, 2010.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[5] S. Dasgupta and V. Ng. Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 701–709. Association for Computational Linguistics, 2009.

[6] S. Fralick. Learning to recognize patterns without a teacher. *IEEE Transactions on Information Theory*, 13(1):57–64, 1967.

[7] L. Gatti, M. Guerini, and M. Turchi. Sentiwords: Deriving a high precision and high coverage lexicon for sentiment analysis. *IEEE Transactions on Affective Computing*, 7(4):409–421, 2016.

[8] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009.

[9] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12, 2009.

[10] Y. He and D. Zhou. Self-training from labeled features for sentiment analysis. *Information Processing & Management*, 47(4):606–616, 2011.

[11] S. Li, Z. Wang, G. Zhou, and S. Y. M. Lee. Semi-supervised learning for imbalanced sentiment classification. In *IJCAI proceedings-international joint conference on artificial intelligence*, volume 22, page 1826, 2011.

[12] Y. Liu, X. Yu, A. An, and X. Huang. Riding the tide of sentiment change: Sentiment analysis with evolving online reviews. *World Wide Web*, 16(4):477–496, jul 2013.

[13] M. Lucas and D. Downey. Scaling semi-supervised naive bayes with feature marginals. In *ACL (1)*, pages 343–351, 2013.

[14] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. Mllib: Machine learning in apache spark. *Journal of Machine Learning Research*, 17(34):1–7, 2016.

[15] S. M. Mohammad, S. Kiritchenko, and X. Zhu. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*, 2013.

[16] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

[17] S. Sedhai and A. Sun. Hspam14: A collection of 14 million tweets for hashtag-oriented spam research. In *SIGIR*, pages 223–232. ACM, 2015.

[18] N. F. F. D. Silva, L. F. Coletta, and E. R. Hruschka. A survey and comparative study of tweet sentiment analysis via semi-supervised learning. *ACM Computing Surveys (CSUR)*, 49(1):15, 2016.

[19] J. Su, J. S. Shirab, and S. Matwin. Large scale text classification using semi-supervised multinomial naive bayes. In *ICML*, pages 97–104, 2011.

[20] P. A. Tapia and J. D. Velásquez. Twitter sentiment polarity analysis: A novel approach for improving the automated labeling in a text corpora. In *International Conference on Active Media Technology*, pages 274–285. Springer, 2014.

[21] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[22] S. Wagner, M. Zimmermann, E. Ntoutsi, and M. Spiliopoulou. Ageing-based multinomial naive bayes classifiers over opinionated data streams. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I*, pages 401–416, 2015.

[23] S. Wang and C. D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, pages 90–94. Association for Computational Linguistics, 2012.

[24] R. Xia, C. Wang, X.-Y. Dai, and T. Li. Co-training for semi-supervised sentiment classification based on dual-view bags-of-words representation. In *ACL (1)*, pages 1054–1063, 2015.

[25] L. Zhao, M. Huang, Z. Yao, R. Su, Y. Jiang, and X. Zhu. Semi-supervised multinomial naive bayes for text classification by leveraging word-level statistical constraint. In *AAAI*, 2016.

[26] M. Zimmermann, E. Ntoutsi, and M. Spiliopoulou. A semi-supervised self-adaptive classifier over opinionated streams. In *ICDM Workshop*, pages 425–432, 2014.