Knowledge and Information Systems
https://doi.org/10.1007/s10115-019-01392-9

REGULAR PAPER

Sentiment analysis on big sparse data streams with limited labels

Vasileios Iosifidis[1] · **Eirini Ntoutsi**[1]

Received: 12 November 2018 / Revised: 29 July 2019 / Accepted: 5 August 2019

Abstract
Sentiment analysis is an important task in order to gain insights over the huge amounts of opinionated texts generated on a daily basis in social media like Twitter. Despite its huge amount, standard supervised learning methods won't work upon such sort of data due to lack of labels and the impracticality of (human) labeling at this scale. In this work, we leverage distant supervision and semi-supervised learning to annotate a big stream of tweets from 2015 which consists of 228 million tweets without retweets (and 275 million with retweets). We present the insights from our annotation process regarding the effect of different semi-supervised learning approaches, namely Self-Learning, Co-Training and Expectation–Maximization. Moreover, we propose two annotation modes, the batch mode where all labeled and unlabeled data are available to the algorithms from the beginning and a lightweight streaming mode that processes the data in batches based on their arrival time in the stream. Our experiments show that stream processing with a sliding window of three months achieves comparable results to batch processing while being more efficient. Finally, to tackle the class imbalance problem, as our dataset is imbalanced toward the positive sentiment class, and its aggravation by the semi-supervised learning methods, we employ data augmentation in the semi-supervised learning process in order to equalize the class distribution. Our results show that semi-supervised learning coupled with data augmentation outperforms significantly the default semi-supervised annotation process. We make the so-called *TSentiment15* sentiment-annotated dataset available to the community to be used for evaluation purposes and for developing new methods.

Keywords Sentiment analysis · Semi-supervised learning · Class imbalance · Data augmentation

1 Introduction

A huge amount of opinions is generated on a daily basis in social media like Twitter and Facebook referring to essentially every entity—products, persons, brands, events or topics. Opinions are valuable for not only consumers, who benefit from the experiences of other

✉ Vasileios Iosifidis
iosifidis@l3s.de

[1] L3S Research Center, Leibniz University Hanover, Hanover, Germany

consumers, in order to make better buying decisions but also for vendors, who can get insights on what customers like and dislike about their products [32]. Such sort of data is freely available nowadays; however, due to their amount and complexity a proper analysis is required in order to gain insights. The analysis of such sort of data is investigated in the areas of sentiment analysis and opinion mining [48,54]. Sentiment analysis aims at characterizing the sentiment content of a text as either positive or negative (some approaches also consider the neutral class).

Traditionally, sentiment analysis is investigated in the fully supervised learning setting, assuming that a fully labeled training set is available, e.g., [36,44–47,63]. However, despite its volume, the amount of labeled data is limited and acquiring (human) labels for all instances at this scale is impractical. Therefore, standard supervised learning methods are not applicable and new methods are required that can exploit both (few) labeled and (huge) unlabeled data for learning a supervised model.

Semi-supervised learning addresses this problem by leveraging unlabeled data, together with the labeled ones, to learn a supervised model. In this work, we employ three well-known semi-supervised learning approaches, Self-Learning, Co-Training and Expectation–Maximization (EM), in order to annotate a huge collection of tweets covering the whole year 2015. Our dataset comprises a stream of instances arriving over time, and therefore, annotation could be implemented either as a batch or a streaming process. The latter is more efficient, and as we show in our experiments, its predictive performance is close to the batch case, when we consider a sliding window of three months.

Semi-supervised learning tackles the problem of label scarcity, but in its basic form, it does not consider class imbalance, which is a very common problem in many applications [23]. In our learning setup, there is a strong imbalance toward the positive sentiment class. The problem of class imbalance in the original dataset is further aggravated via the semi-supervised learning process. To this end, we propose to integrate data augmentation techniques in the semi-supervised learning process to rebalance the classes, and we investigate different forms of data augmentation that are applicable to this domain.

We summarize our findings from the annotation process which cover a variety of interesting aspects, and we make our annotations available to the community in an attempt to provide more complex datasets with temporal characteristics that can facilitate further research in the areas of sentiment analysis and data stream mining, in general. Our contributions are summarized below:

- We label a big Twitter stream dataset with limited labels consisting of 228M tweets without retweets and 275M tweets with retweets; the collection spans the whole year 2015 and is therefore also appropriate for temporal analysis. We make our labels available to the community to facilitate the development of new methods for sentiment analysis and stream mining in general and for evaluation purposes.
- We extensively evaluate the performance of different semi-supervised learning approaches, namely Self-Learning, Co-Training and EM, and how it is affected by the amount of labeled data, the amount of unlabeled data and the classifier's confidence threshold.
- We process the data in two different modes: i) as a batch, where labeled and unlabeled data are available to the algorithm from the beginning, and ii) as a stream, where both labeled and unlabeled data are gradually available to the algorithm as the stream progresses. We show that the latter approach with a sliding window of three months achieves a comparable to the batch approach accuracy, while being more efficient.

– We employ data augmentation with semi-supervised learning in order to tackle the problem of class imbalance that exists in the original dataset, and it is further aggravated by the iterative semi-supervised learning approaches. We show that augmentation can help in tackling the class imbalance problem.
– We report on the impact of data redundancy (via retweets) in the performance of the different models. As retweets can be considered as a natural form of data augmentation, we also report on their impact on class imbalance.
– We provide a qualitative evaluation of our labeling process via crowd sourcing.

This work is an extension of our previous work [26]. The major changes include: i) including EM in the evaluation as an example of semi-supervised learning method with soft assignments, ii) tackling the problem of class imbalance in the semi-supervised learning process via data augmentation, iii) qualitative evaluation of the derived labels via crowdsourcing and iv) comparison of our annotations to state-of-the-art sentiment annotation tools.

The rest of the paper is organized as follows: Related work is presented in Sect. 2. In Sect. 3, we describe our dataset and how we derived the ground truth for learning. The semi-supervised learning approaches are described in Sect. 4. In Sect. 5, we describe the augmentation process to handle class imbalance. Our experiments for batch and stream annotation, crowd-source evaluation and comparison to state-of-the-art methods are described in Sect. 6. Finally, conclusions and outlook are presented in Sect. 7.

2 Related work

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

2.1 Sentiment analysis

Due to the abundance of opinionated texts, there is a lot of research on sentiment analysis regarding the effect of different learners, building domain-specific learners and transferring across different domains. For example, Pang et al. [47] examine the effectiveness of machine learning algorithms such as naive Bayes classifiers, maximum entropy models and support vector machines to the problem of sentiment classification. In a follow-up work [46], the authors investigate the rating-inference problem for which instead of classifying reviews as positive or negative they try to determine the score w.r.t. a multi-point scale. Melville et al. [36] propose a framework for domain-specific sentiment classification that employs lexical information with a model trained upon a given corpus. A similar approach is proposed by Melidis et al. [34] but for temporal collections. Ye et al. [63] classify reviews from travel blogs using naive Bayes and SVMs combined with a character-based N-gram model. Paltoglou and Thelwall [44] employ weighting schemes from information retrieval such as *tf-idf* to improve the classification accuracy on sentiment classification tasks. Pan et al. [45] investigate sentiment classification across different domains by combining domain-specific words from different domains. Additionally, Hube and Fetahu [25] propose an approach that determines phrasing bias, which detect the use of subjective language toward specific events or other entities in terms of words or phrases that are either one sided or inflammatory.

More recently, there is also a lot of work on sentiment analysis over temporal data and data streams. For example, Iosifidis et al. [27] propose a classification framework for opinionated streams which adapts to concept drifts that are detected as vocabulary changes over a sliding

window. Melidis et al. [35] consider the problem of sentiment classification under feature drifts, where different features might undergo different types of temporal drifts and propose an ensemble of different experts, each specialized to capture a particular trend. Unnikrishnan et al. [58] propose learning entity-specific models, instead of a global model to facilitate detecting local changes that are not always reflecting in the global stream.

2.2 Semi-supervised learning

Semi-supervised learning addresses this problem by leveraging unlabeled data, together with the labeled data, to learn classification models. Nigam et al. [41,42] propose an algorithm for learning from labeled and unlabeled documents based on Expectation–Maximization and multinomial 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 documents. The process is repeated until convergence. This basic algorithm was improved by two extensions: by employing a weighting factor to modulate the contribution of unlabeled data and by using multiple mixture components per class, instead of a single one. Su et al. [55] propose a semi-supervised extension of MNB, called 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 parameters of an MNB classifier. Lucas and Downey [33] introduced MNB-FM a method that extends MNB to leverage marginal probabilities of the words, computed over the unlabeled data. The marginal probabilities are used as constraints to improve the class-conditional probability estimates for the positive and negative classes. Zhao et al. [66] proposed MNB-WSC, which preserves reliable word estimates, as extracted from a sufficient amount of labeled data. We also use MNB as our base model. Despite its class-conditional feature independence assumption, MNB is known to perform moderately, and in some cases, it has been reported that its performance for short texts is equal or superior to more complex models [61].

A comprehensive survey of semi-supervised learning approaches for Twitter is provided in this recent survey [52]. Similarly to us, they found that Co-Training performed better with limited labels, whereas Self-Learning 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 and Ng [10] 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 Aue and Gamon [1], where the authors exploit a small number of labeled and a huge amount of unlabeled data using EM.

Self-Learning is categorized as a form of semi-supervised learning [17]. The central idea behind Self-Learning is that we can expand the training set by using the most confident predictions of the classifier. Self-Learning might cause error propagation as the predictions of the classifier are then used for its training [24,68]. To deal with these issues, Co-Training was introduced in Blum and Mitchell [8] that combines two different views of the data in two classifiers, which are then used together 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 Li et al. [30], the authors study class imbalance for semi-supervised classification. They use undersampling 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. Xia et al. [62] use negations and antonyms to generate an opposite view of the training data; the original and the opposite view are exploited afterward via Co-Training.

2.3 Large-scale annotation

TSentiment [20] is a dataset of 1.6 million tweets covering the period from April 6, 2009, to June 25, 2009, which are annotated as positive or 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 to its temporal aspects [6,60]. TweetsKB [15] is another interesting dataset, which contains entity-related annotations such as co-entities, popularity and also includes sentiment annotations via SentiStrength tool [29]. The dataset can be exploited to analyze social media archives e.g., Fafalios et al. [16]. Finally, HSpam14 [51] 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 remaining unlabeled tweets. Our dataset covers a larger period, of one year and therefore is more appropriate for such tasks.

3 Dataset description

The dataset and the different preprocessing steps as well as their effect on the dataset are described in Sect. 3.1. In Sect. 3.2, we describe how we derive the training set, i.e., labeled instances for the classification task. In Sect. 3.3, we provide an exploratory analysis of the dataset with a focus on its temporal characteristics.

3.1 Data collection and preprocessing

The dataset has been collected[1] from the 2015 Twitter stream using its public streaming API,[2] which provides a random selection of tweets (about 1% of all tweets). 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 retweets (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 that are described below:

- *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 in 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 latter, we replaced the negation with its antonym, e.g., "not bad" → "good," using WordNet list.[4]

[1] The Twitter crawling collection project is part of the L3S research center initiative.

[2] https://dev.twitter.com/streaming/overview.

[3] www.noslang.com.

[4] https://wordnet.princeton.edu/.

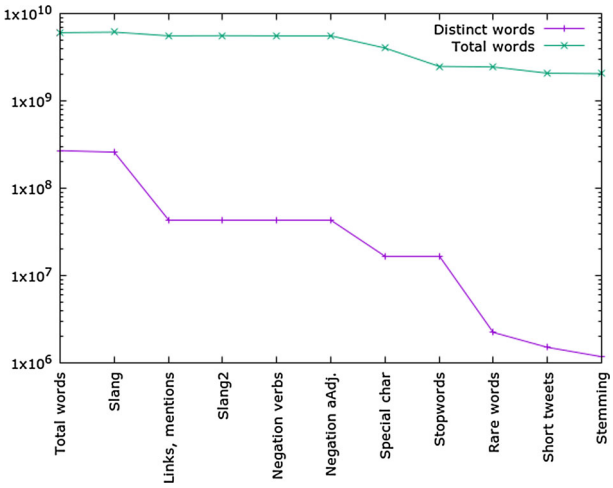


Fig. 1 Preprocessing effects

- *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 the 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., Sect. 3.2).
- *Stopword removal* We removed stopwords using Weka's stopwords list.[5]
- *Stemming*: We applied Porter stemmer.
- *Removal of rare words* We removed rare words from the corpus, using a frequency of 10 as the cutoff value.
- *Removal of short tweets* Finally, we removed tweets with less than four (< 4) words after the aforementioned steps, similarly to [56].

The preprocessing resulted in a reduction of the corpus and of the vocabulary (i.e., distinct words) (c.f., Fig. 1). In particular, the corpus was reduced by 41% (from 384M to 228M tweets) and the vocabulary was reduced by 99,5% (from 269M to 1,17M distinct words). Figure 2 shows the frequency distribution of the unique words in the corpus, a total of 1.6B words. As we see, the majority of unique words has less than 100 occurrences (almost 1M words). Around 150K unique words are occurring from 100 to 1K times, while only 18K unique words are occurring more than 1K times.

3.2 Building the ground truth for learning

In the absence of human labels, we derive our ground truth for learning by combining two approaches/experts: i) an emoticon-based approach and a sentiment lexicon approach (using SentiWordNet.[6]) Our idea is to consider as ground truth those tweets for which both experts agree in their labelings.

[5] http://weka.sourceforge.net/doc.dev/weka/core/stopwords/Rainbow.html.

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

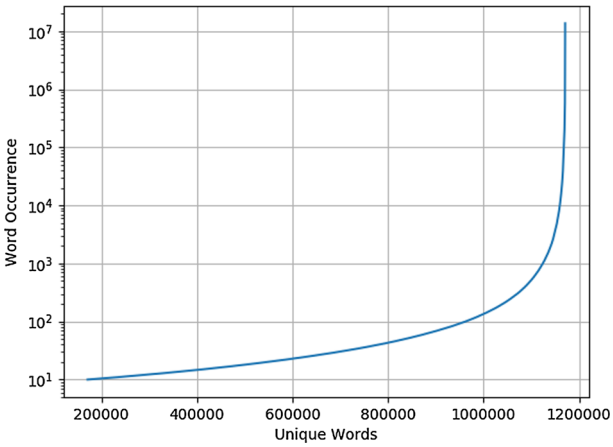


Fig. 2 Frequency distribution of unique words

Deriving labels from emoticons Using a list of positive[7] and negative[8] 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 [20].

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 5.3M (= 3.8M+1.5M) 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 [57] and considering 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 [18]. A word's sense is associated with a score that is calculated by computing a weighted average of the differences between the positivity and negativity scores assigned to the various senses of the word. Same as in Berardi et al. [5], we employ harmonic mean function (1) to aggregate the different senses of a given word since the senses are sorted according to frequency in descending order [18].

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} \tag{1}$$

Building the ground truth The results of juxtaposing the emoticons- and SentiWordNet-based labels are presented in Table 1. We considered as our ground truth the true positives, i.e., tweets where both emoticon-based and SentiWordNet-based labeling agree. SentiWordNet supports also neutral class as shown in Table 1; however, due to the inability of emoticon-based approach to distinguish efficiently the neutral class, we employ only the positive and negative class tweets hereafter. Our final ground truth dataset consists of 2,527,753 tweets.

[7] Positive emoticons: $c) =] :] :\} ; > :>) :^) : D =) ;) :) 8) (: (; : o) : -) : P < 3 : 3 \; ^\_^$

[8] Negative emoticons: $-c : [: \{ :< : -(: / : -[: c : - < : (\; :' \{ >: [$

Table 1 Emoticon versus SentiWordNet-based labelings

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

Agreements marked in boldface

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

An interesting observation from Table 1 is the disagreement between the two labeling experts: distant supervision using emoticons and SentiWordNet. A profound reason is the existence of the extra neutral class in case of SentiWordNet. However, the sources of disagreement extend beyond this; in particular, SentiWordNet is a static database which has been generated upon WordNet Gloss Corpus,[9] a corpus of manually annotated WordNet synset definitions. Such a database does not capture the large variability of words in a social stream (due to, e.g., the creation of new medium-specific words like hashtags); in fact, the coverage is pretty narrow [34]. On the other side, SentiWordNet contains high-quality sentiment annotations comparing to the distant supervision approach that relies on emoticons as proxies for sentiment, and therefore, it is prone to errors.

3.3 Temporal characteristics

The temporal distribution of our dataset is depicted in Fig. 3 including both ground truth tweets (c.f., Sect. 3.2) and unlabeled ones. As we can see, the amount of unlabeled tweets is vast compared to the amount of labeled ones 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.

3.4 Comparing ground truth to SentiStrength and TreeBank

For our derived ground truth, we also provide a comparison with state-of-the-art sentiment analysis methods such as SentiStrength [29] and Sentiment TreeBank [53].

SentiStrength [29] has been trained on social web data and can predict the sentiment (positive/negative) of short texts. In detail, SentiStrength has been trained upon 2600 human-classified comments from the social network Web site "myspace.com," which were extracted in December 2008. This method assigns both positive and negative scores in the range $+1$ to $+5$ for the positive class and -1 to -5 for the negative class. For our comparison, we consider as neutral those tweets where both positive and negative scores are the same (in absolute values).

Sentiment TreeBank [53] has been trained upon 11,855 sentences which were extracted from movie reviews in 2005 from the "rottentomatoes.com" Web site [46]. TreeBank is able to classify sentences of arbitrary lengths into five classes: "negative," "somewhat negative," "neutral," "positive" and "somewhat positive." For our comparison, we merge "negative"

[9] http://wordnetcode.princeton.edu/glosstag.shtml.

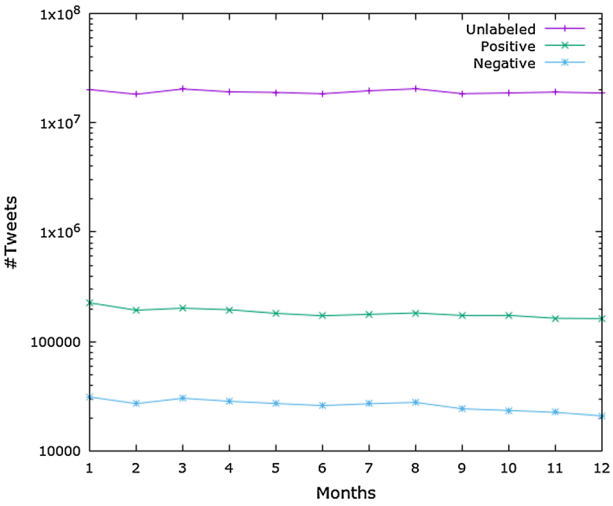


Fig. 3 Dataset distribution on a monthly basis

Table 2 SentiStrength versus ground truth labelings

SentiStrength	Ground truth	
	Positive (%)	Negative (%)
Positive	**81.18**	22.43
Negative	1.39	**53.97**
Neutral	17.44	23.61

Agreements marked in boldface

Table 3 TreeBank versus ground truth labelings

TreeBank	Ground truth	
	Positive (%)	Negative (%)
Positive	**28.26**	5.69
Negative	45.70	**75.48**
Neutral	26.04	18.83

Agreements marked in boldface

and "somewhat negative" classes into the negative class as well as "positive" and "somewhat positive" classes into the positive class.

In Tables 2 and 3, we compare SentiStrength and TreeBank to our ground truth. SentiStrength has a higher agreement w.r.t. the positive class (81.18%)—the agreement w.r.t. the negative class is much smaller (53.97%) though it still comprises the majority agreement class. For TreeBank, the situation is inverse; the agreement on the negative class is the strongest (75.48%), whereas the agreement in the positive class is much smaller (28,26%) and even worse; most of the positive ground truth was labeled as negative in TreeBank.

As we can see, both methods exhibit high disagreement w.r.t. our ground truth labels. This is caused due to the different training data employed by the state-of-the-art methods, the different media/application domains from which the data are collected as well as the different spanning periods of each dataset. Moreover, we do not consider the neutral class, in contrast to TreeBank and SentiStrength.

4 Sentiment learning with limited labels

In our learning setup, we have a small number of labeled instances, denoted by L, and a huge number of unlabeled instances, denoted by U. This is a typical setup in many real-life applications as despite the huge amounts of data nowadays, only a small fraction of these data are labeled and can be therefore directly used for (supervised) learning. To deal with this issue, semi-supervised learning approaches exploit both labeled and unlabeled data for training—the underlying assumption is that having access also to the unlabeled data allows us to better understand the underlying class distribution [67]. In this work, we investigate three popular semi-supervised learning approaches: Self-Learning, Co-Training and EM described hereafter.

4.1 Self-Learning

The main idea of Self-Learning [17] is to use the labeled set L to build an initial classifier, then iteratively apply the model to the unlabeled corpus U and, in each iteration, expand the training set L with instances from the unlabeled corpus which were predicted with high confidence by the classifier; the confidence is evaluated according to a user-defined threshold δ. The pseudocode of the Self-Learning algorithm is shown in Algorithm 1.

The initial training set T is the labeled set L (line 1). In each iteration, the training set is expanded by including confident predictions from U (lines 5–8); the expanded training set is used for building a new classifier (line 3). The procedure continues until the stopping criterion is met e.g., when U is empty or after a certain number of iterations, or if no further expansion is possible due to the threshold δ.

Algorithm 1: Pseudocode of Self-Learning

```
   Input: L: labeled set, U: unlabeled set, δ: confidence threshold
   Result: T: labeled set
1  T ⟵ L
2  while (stopping criterion) do
3  |  Φ ⟵ train classifier on T;
4  |  for i=1 to |U| do
5  |  |  if (confidence of Φ.classify(U_i) ≥ δ) then
6  |  |  |  T ⟵ T ∪ U_i, where U_i is the i-th instance in U
7  |  |  |  Mark U_i as labeled;
8  |  |  end
9  |  end
10 |  Update U by removing labeled instances;
11 end
12 return T;
```

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 [24,68]. 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 [8] assumes that the feature space, lets denote it by X, can be split into two parts, $X = (X^1, X^2)$, the so-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 initial labeled set L, but each on its own view, $X_i, i \in \{1, 2\}$. In the original Co-Training approach [8], the unlabeled data are used to expand the joint training set as follows: At each iteration, Φ^1 classifies a few unlabeled instances for which it is more confident about and appends them to the joint training set. Similarly for Φ^2. The updated training set is then used for building the two new classifiers, again, each classifier is trained on its own view.

Algorithm 2: Pseudocode of Co-Training

Input: L: labeled set, X^1, X^2: the two feature views, U: unlabeled set, δ: confidence threshold
Result: T_1, T_2: labeled sets

1 $T_1, T_2 \longleftarrow L$;
2 $U^1, U^2 \longleftarrow U$;
3 **while** *(stopping criterion)* **do**
4 $\quad \Phi_1 \longleftarrow$ train classifier on T_1 (using X_1 view);
5 $\quad \Phi_2 \longleftarrow$ train classifier on T_2 (using X_2 view);
6 $\quad TempSet_1 = \emptyset$;
7 $\quad TempSet_2 = \emptyset$;
8 $\quad$ **for** *i=1* **to** $|U^1|$ **do**
9 $\quad\quad$ **if** *(confidence of* Φ_1*.classify(*U_i^1*)* $\geq \delta$*)* **then**
10 $\quad\quad\quad TempSet_1 \longleftarrow TempSet_1 \cup U_i^1$, where U_i^1 is the i-th instance in U^1
11 $\quad\quad\quad$ Mark U_i^1 as labeled
12 $\quad\quad$ **end**
13 $\quad$ **end**
14 $\quad$ **for** *i=1* **to** $|U^2|$ **do**
15 $\quad\quad$ **if** *(confidence of* Φ_2*.classify(*U_i^2*)* $\geq \delta$*)* **then**
16 $\quad\quad\quad TempSet_2 \longleftarrow TempSet_2 \cup U_i^2$, where U_i^2 is the i-th instance in U^2
17 $\quad\quad\quad$ Mark U_i^2 as labeled
18 $\quad\quad$ **end**
19 $\quad$ **end**
20 $\quad$ Update U^1 and U^2 by removing labeled instances;
21 $\quad T_1 \longleftarrow T_1 \cup TempSet_2$;
22 $\quad T_2 \longleftarrow T_2 \cup TempSet_1$;
23 **end**
24 return T_1, T_2;

We follow a slightly different version (c.f., Algorithm 2) by maintaining a different training set for each classifier. We initialize Co-Training as above, with Φ^1, Φ^2 classifiers built upon the initial 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 and unlabeled set U (lines 1–2), over the iterations and as one learns from the other, the training sets of the two classifiers are different (lines 20–22). The procedure stops when the stopping criterion is met e.g., when U^1 or U^2 is

empty or after a certain number of iterations, or if no further expansion is possible due to the threshold δ.

The intuition behind Co-Training is that each classifier provides labeled data to the other classifier, which the latter 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 while preventing each classifier to propagate its own error, as in Self-Learning. Two assumptions are proposed for Co-Training to work well: First, that each view (classifier) is able to learn the target concept given enough data, that is, each view is *sufficient* for learning and, second, that the views are conditionally independent, that is, the two views are independent given the class. Theoretical results have shown that if the sufficiency and independence assumptions are satisfied, Co-Training is guaranteed to work; however, verifying that these assumptions hold in real datasets is not straightforward [13]. Luckily, however, Co-Training has been successful in many real-world tasks even if the aforementioned conditions could not be ensured, e.g., Nigam and Ghani [40].

4.3 Expectation–Maximization (EM)

Expectation–Maximization (EM) belongs to a class of algorithms that iteratively estimates the maximum a posteriori probabilities in statistical models by exploiting the structure of the data (labeled and unlabeled) [11]. The algorithms consist of two steps: the Expectation- or E-step and the Maximization- or M-step. EM starts by initializing model's parameters based on the labeled data (expected distribution, E-step). Afterward, new instances are revealed to the model for which it tries to fit the current probability distribution to include the new data (M-step). These two steps are repeated until the stopping criterion is met e.g., maximum number of iterations is achieved or the current distribution does not change from the E-step to the M-step (convergence).

The EM pseudocode is shown in Algorithm 3. A classifier is trained upon the initial labeled data L (line 2). Afterward, the classifier is employed to assign probabilistically weighted class labels to the unlabeled data U (lines 4–7). Then, the classifier is rebuilt upon the labeled data and the unlabeled data which has been labeled by the previous classifier (line 8). The procedure is repeated until the stopping criterion is met (lines 3–9). The final classifier Φ is used for final labeling of the unlabeled data U (lines 10–12).

The algorithm guarantees improvement of a parameter's estimation through each iteration if the mixture model assumption holds [11]. If the model is wrong though, the unlabeled data may actually hurt the performance [9]. Moreover, EM is prone to local maxima, and if a local maximum is far from the global maximum, unlabeled data might again hurt the performance of the learner [43].

4.4 Discussion

We employ three well-known semi-supervised methods to exploit the unlabeled data U together with the labeled data L in order to obtain more accurate predictions. All methods use unlabeled data to modify hypotheses obtained from labeled data alone. However, each method comes with its own advantages and limitations. Self-Learning is an easy-to-use algorithm, but it can propagate errors in the predictions which affect the next iterations and eventually the final result. Co-Training can overcome to some extent this problem by relying on two different learners that train each other by providing confident predicted labels.

Algorithm 3: Pseudocode of EM

```
Input: L: labeled set, U: unlabeled set
Result: T: labeled set
T = ∅;
Φ ⟵ train classifier on L;
while (stopping criterion) do
    for i=1 to |U| do
        Φ.classify(U_i), where U_i is the i-th instance in U
        Mark U_i as labeled;
    end
    Φ ⟵ train classifier on L ∪ U;
end
for i=1 to |U| do
    Φ.classify(U_i);
    T ⟵ T ∪ U_i;
end
return T;
```

However, Co-Training works well under the sufficiency and independence assumption which do not always hold for real-world datasets. Finally, EM with generative mixture models tries to maximize the likelihood estimates of a model's parameters based on the data. However, it may perform poorly if the assumption on the correlation between classes and model components is violated and it might get stack to some local maximum. Moreover, EM can be extremely slow when data are multi-dimensional. Finally, both Self-Learning and Co-Training make hard assignments, whereas EM is the only method that probabilistically assigns an instance to all classes (soft assignment).

5 Overcoming class imbalance via data augmentation

Except for label scarcity, our learning setup is also characterized by class imbalance. In particular, the positive class is constantly overrepresented over the stream comparing to the negative sentiment class (c.f., Fig. 3). Models trained upon imbalanced data learn mainly the majority class while ignoring the minority [23]. In our case, the problem is aggravated due to the propagation of the predicted labels in the next rounds of the semi-supervised learning process. As a result, the tendency of the models toward the majority class is much higher in the final models, as we also show in our experiments (c.f., Sect. 6.6).

Traditionally, class imbalance is handled through oversampling (from the minority class) and/or undersampling (from the majority class). Both approaches, however, come with limitations [12,14]; in undersampling, one cannot control what information about the majority class is thrown away, whereas in oversampling, no new information is added to the training set, rather some instances from the minority class are replicated, thus having a stronger effect on the classifier. In this work, except for oversampling and undersampling, we also employ data augmentation techniques for generating plausible pseudo-instances for the minority class in order to correct for the class imbalance.

Augmentation is integrated into the semi-supervised learning process in an attempt to control the class imbalance problem over the training iterations. An overview of our approach is depicted in Fig. 4. The training set is rebalanced using data augmentation (Sect. 5.1) and the semi-supervised learning process takes place as before but over the balanced data. We consider the labeled set balanced when the difference between positive (P_+) and negative

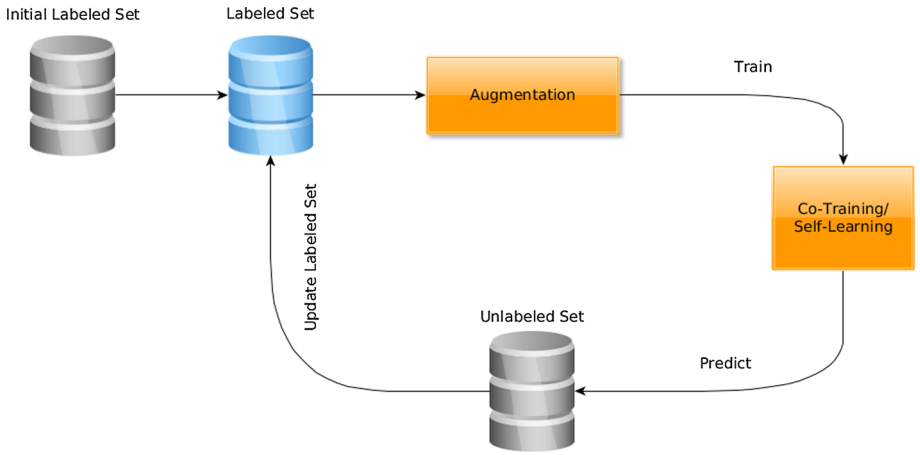


Fig. 4 Augmentation-assisted semi-supervised learning

(P_-) instances is smaller than a user-defined class-balance threshold ϵ (in our experiments, we set $\epsilon = 20\%$).

5.1 The data augmentation process

Given an original binary classification dataset D with class imbalance (without loss of generality, let us assume that the negative class is the minority, i.e., $|P_-| << |P_+|$ for which $|P_-|$ is the amount of negative instances and $|P_+|$ the amount of positive instances), our goal is to build a balanced dataset D' via data augmentation, i.e., to generate new instances out of the original instances by applying domain-specific, label-preserving transformations. We refer to the generated instances as *pseudo-instances* and to D' as *augmented dataset*. Note that $D \subset D'$.

We propose two augmentation techniques: semantic augmentation (Sect. 5.1.1) and blank-out corruption (Sect. 5.1.2) for balancing the classes, beyond the well-known undersampling and oversampling approaches (Sect. 5.1.3). The former employs semantic similarity between words (via word embeddings) in order to create semantically similar instances out of the original instances. The latter corrupts the training instances by removing information (words) from the original instances and thus creates corrupted versions of the original instances. Both techniques are *feature transformation* techniques, i.e., they change the individual features/words. In order to ensure that the augmented instances will preserve their labels, we "transform" only non-sentimental words.

The words with a score higher than zero are considered as sentimental words, the rest as non-sentimental. For our dataset, 38,222 words were found to be sentimental while the remaining terms (1,132,729) were filtered out.

5.1.1 Data augmentation via semantic similarity

To generate pseudo-instances of the same class, we employ the semantic similarity of words as captured through word embeddings [38]. Our idea is to generate pseudo-instances by replacing words in the original document with semantic similar words. In particular, for a

selected word w occurring in an original document d, we generate its similar words based on their embedding vectors and we select randomly one of the top-k similar words w' to replace the original w in the pseudo-instance d'. We only consider words which are sentimental. As an example, the text "I love this car very much" could generate "I like this car very much."

There exist different word embedding versions based on the data used for their training. In this work, we employ Glove word embeddings [49] which have been generated by 2B tweets. For each sentimental word in our corpus, we generate the top-k most similar words based on Glove embeddings (in our experiments, we set $k = 10$). A list of top-k similar words, called *similarity list*, is generated from the aforementioned process which contains 33,037 terms (the remaining 5 thousand words were not included in Glove embeddings).

Given an instance d, the procedure for generating a pseudo-instance d' out of it is as follows: For the sentimental words $w \in d$, check whether w exist in the similarity list. If not, ignore w. Otherwise, replace w with a randomly selected similar word from its top-k semantic similar words according to Glove.

5.1.2 Data augmentation via corruption

Corruption of images via noise is a common transformation in the image domain and aims at building more robust machine learning models [19]. We follow a similar idea for text: we generate pseudo-instances by deleting a (randomly selected) word from the original document. To ensure that the class label is preserved, we do not remove negations or sentimental words (c.f. Sect. 5.1 on sentimental words).

To ensure that the resulting document is still plausible, we apply corruption to sufficiently long documents, namely to documents of at least four words as suggested by Tapia and Velásquez [56]. As an example, the text "I don't like the morning traffic" could be transformed into "I don't like the traffic." Since our target is class imbalance, we generate pseudo-instances only for the minority class.

5.1.3 Oversampling and undersampling

Oversampling (shortly, Over.) and undersampling (shortly, Under.) do not operate on the feature level but rather on the instance level. Oversampling is repeatedly applied on the minority instances, by randomly duplicating instances, until the difference between positive and negative instances is less or equal to the class-balance user-defined threshold ϵ. Undersampling is applied on the majority class by randomly removing instances from the majority class until the threshold ϵ is met.

Moreover, we used a combination of (random) oversampling and undersampling (shortly, Over. and Under.) that works as follows: (i) First undersampling is applied by removing half of the majority's instances (b) if there is still class imbalance (according to the user-defined class-balance threshold), oversampling is applied on the minority instances until the threshold is met, otherwise the process terminates.

5.1.4 Discussion

Augmentation techniques like semantic similarity and corruption have a similar goal to oversampling/undersampling, namely to balance the population of the two classes. There are, however, fundamental differences between the different approaches, and each one comes with its own assumptions and limitations. On the one hand, oversampling does not add any

new information to the process and can also amplify existing noise by duplicating noisy instances. On the other hand, undersampling may result in removing valuable information from the dataset which can degrade the overall performance of the model. Semantic augmentation depends on the quality of the pre-trained word embeddings. If the employed word embeddings come from a different corpus than the applied, then the dictionary intersection between word embeddings and the corpus will be limited. In addition, polysemous words may totally change the context of a sentence; for example, "I like apple products" which refers to the famous company can be converted to "I like vegetable products." Corruption can also change the sentiment of a sentence; for example, "I support banning smoking in public areas" can be converted to "I support smoking in public areas." Finally, a common pitfall in all augmentation methods is that by augmenting already noisy instances the reinforcement of noise and mistakes is inevitable, and therefore, the overall data quality is degraded.

6 Experiments

We report on the Twitter dataset introduced in Sect. 3. We employ multinomial naive Bayes (MNB) as our basic classifier for Self-Learning, Co-Training and EM. We choose naive Bayes as our base learner due to its ability to handle huge amounts of features, while being tolerant to irrelevant features. Moreover, due to its simplicity, it can be trained upon vast amounts of training data extremely fast [41,42], compared to, e.g., neural networks such as [53] requiring up to 5h for training for small-scale datasets (less than 10K instances). Finally, it can cope with dynamic feature spaces and errorless model update, both important properties for our stream evaluation. For the implementation,[10] we have used Spark's distributed environment (version 1.6) and its machine learning library MLlib [37].

Co-Training requires two different feature spaces. Therefore, except for the unigrams, we also experimented with two different feature sets: (i) bigrams: The feature space was extracted from the preprocessed text, as described in Sect. 3, (ii) language features: We extracted moreover POS tags using Stanford's POS tagger [57] as well as syntactic features like number of words in capital, words with repeated characters, links and mentions. Moreover, we employed a dictionary which contained sentimental hashtags [39] and counted the occurrences of positive and negative hashtags in our tweets, if any (the extraction was done over the original tweets, not the preprocessed ones). We refer to this feature space as "SpecialF."

Therefore, we have two alternative feature setups for Co-Training:

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

These *views* are not conditionally independent. Nonetheless, recent works indicate that relaxation of the independence criteria does not have much impact on the performance [3,7,31,65]. For the EM and Self-Learning approaches, we used unigrams (after preprocessing).

Our evaluation examines the performance of the different semi-supervised learning approaches w.r.t. the following aspects:

- batch versus stream annotation setup (Sect. 6.1, 6.2, respectively)
- effect of augmentation on class imbalance (Sect. 5)
- effect of redundancy (retweets) on performance (Sect. 6.5)
- a qualitative evaluation of the derived labels based on crowd sourcing (Sect. 6.3).

[10] Source code and data are available at: https://iosifidisvasileios.github.io/Semi-Supervised-Learning/.

6.1 Performance of batch annotation

We split our ground through (2.5 million tweets) in tenfold, 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, and 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 tenfold cross-validation in terms of classification accuracy for Self-Learning and Co-Training, under different confidence thresholds δ that determines which of the classifier predictions are incorporated in the training set. For EM, we do not set any threshold since it is the algorithm's property to maximize the data likelihood based on the predictions.

6.1.1 Self-Learning-based batch annotation

In Fig. 5 (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. 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 L); 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 are 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 Fig. 5 (bottom). At $\delta = 65\%$, for example, the expanded training set is about 8100% 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 addition of the ne predictions does not 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, only a few predictions can satisfy it.

The annotated dataset is depicted in Table 4: 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 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 percentage 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 expanded with more examples of the positive class.

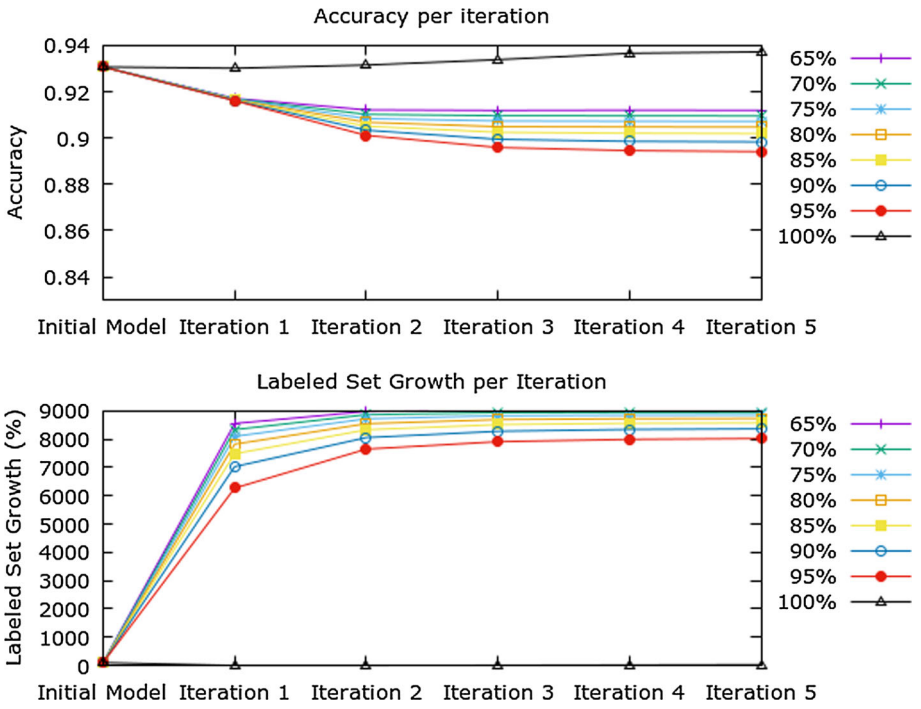


Fig. 5 Batch annotation with Self-Learning: accuracy and labeled set growth under different δ values while the algorithm iterates through the remaining unlabeled tweets

Table 4 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%)	

6.1.2 Co-Training-based batch annotation

Figure 6 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 is very small.

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

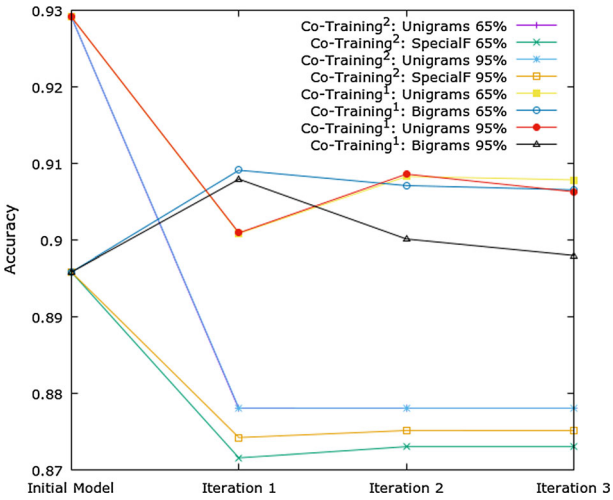


Fig. 6 Batch annotation with Co-Training: accuracy for $\delta = 65\%$, $\delta = 95\%$ while the algorithm iterates through the remaining unlabeled tweets. The accuracy is displayed for each Co-Training classifier member

Table 5 Average accuracy of Co-Training^{1}[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

sifier for the comparison, and we refer to it as Co-Training. We show the accuracy of this model under different thresholds δ in Table 5. 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 $\delta = 100\%$ which is not affected that much as the training set is not much expanded due to the very selective δ.

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 Fig. 7 (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 6. Similarly to what we have observed for Self-Learning, the more selective δ is, the more tweets remain unlabeled; at $\delta = 100\%$, almost all tweets (99.44%) remained unlabeled. Moreover, the amount of unlabeled tweets is smaller comparing to the Self-Learning in Table 4. Regarding the class distribution of the predictions,

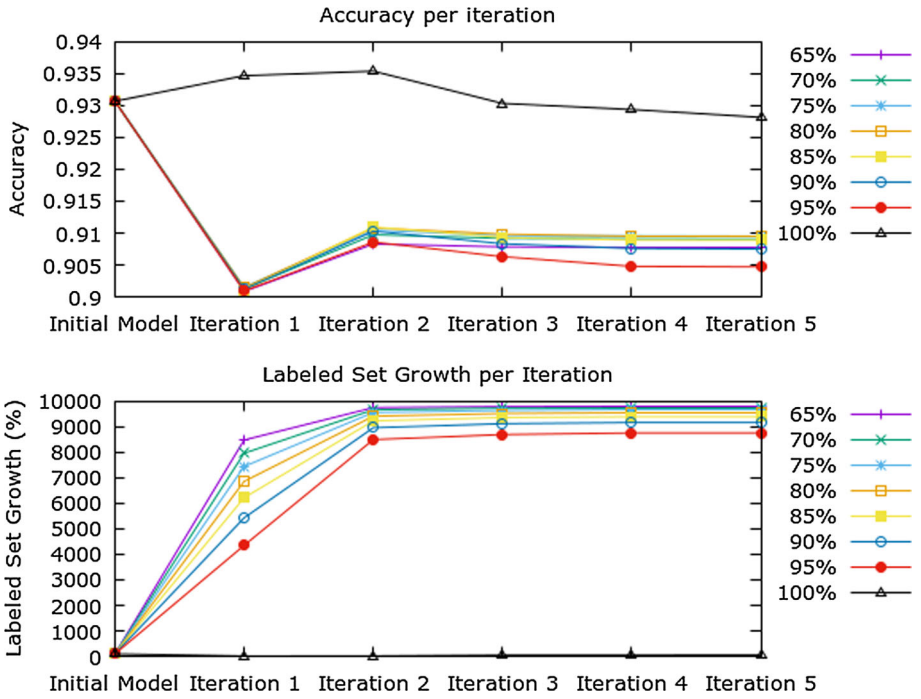


Fig. 7 Batch annotation with Co-Training: accuracy and labeled set growth under different δ values while the algorithm iterates through the remaining unlabeled tweets

Table 6 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%)	5116 (0.39%)	99.44
Initial model	2.211.091 (87.47%)	316.662 (12.52%)	

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

6.1.3 EM-based batch annotation

In Table 7, we report the accuracy of EM per iteration. We stop at 5 iterations as after that the log likelihood does not change too much. The accuracy of EM drops significantly as the

Table 7 Batch annotation with EM: annotated results per class over the iterations

Iter.	Accuracy (%)	Positive predictions	Negative predictions
1	93.52	177,770,034 (77.03%)	53,023,155 (22.97%)
2	90.87	177,559,316 (76.93%)	53,233,873 (23.07%)
3	87.83	176,702,013 (76.56%)	54,091,176 (23.44%)
4	85.31	175,936,532 (76.23%)	54,856,657 (23.77%)
5	83.53	175,162,259 (75.90%)	55,630,930 (24.10%)

number of iterations increases. In particular, starting with a good initial model (93.52% in the first iteration trained on only labeled data), it drops to 83.53% in the fifth iteration trained upon all data. A possible reason is that the unlabeled data alter the model in a way that hurts the overall performance. Note here that in contrast to Self-Learning and Co-Training we do not use only qualified instances for the dataset expansion (based on some threshold δ), rather we accept all instances.

We also observe that the classifier predictions become more balanced over the iterations with the amount of predicted negative instances growing by 210K from the first to second iteration and by 860K instances from second to third iteration.

6.1.4 Comparison of EM, Self-Learning and Co-Training

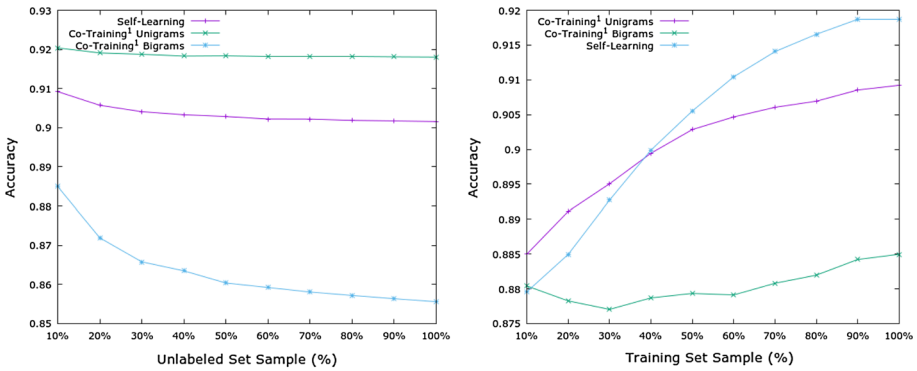
In Table 8, we report the average (over all iterations) accuracy of Co-Training and Self-Learning for different δ values (for the Co-Training, we report here on the performance of the best classifier member, i.e., the one built upon unigrams). As we can see, Self-Learning accuracy decreases (in absolute numbers) with δ much faster than the accuracy of Co-Training. As we can see from Tables 4, 6, Co-Training produces more labels than Self-Learning and results in better class balance.

By comparing Tables 4, 6 and 7, we observe that Self-Learning produces more imbalanced outcomes as δ increases compared to Co-Training and EM. Since the initial ground truth is already highly imbalanced, Self-Learning propagates this behavior stronger than Co-Training and EM, as it is the most sensitive to its own errors method. As δ increases, Self-Learning trains progressively upon its most confident predictions, which in the majority are positive instances. As we show in a later section (c.f., Sect. 6.6), Self-Learning propagates more positive errors compared to Co-Training, as δ increases. Co-Training also produces imbalanced outcomes as δ increases, however, at a much lower rate compared to Self-Learning. In Co-Training, the models may not propagate their own errors; however, they are still subject to errors; in our case, such errors might occur due to, e.g., the fact that unigrams and bigrams do not provide completely independent feature spaces, and therefore, they might fall in the same pitfall, as Self-Learning, for high δ values. EM, on the other hand, is not bound to threshold (δ) scores (in this scenario), and thus, it maintains a similar class ratio to the initial ground truth dataset.

Thus far, we expand the training set based on the confidence threshold δ, which, however, results in an uncontrolled expansion (in terms of number of annotated 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 the annotated set, we randomly select [10–100%] predictions/instances which we then use for dataset expansion. The results are depicted

Table 8 Batch: Self-Learning versus 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	90.49	90.78
90	90.31	90.69
95	90.03	90.50
100	93.38	93.16
Initial model	93.07	93.07



(a) Unlabeled set expansion **(b)** Labeled set expansion

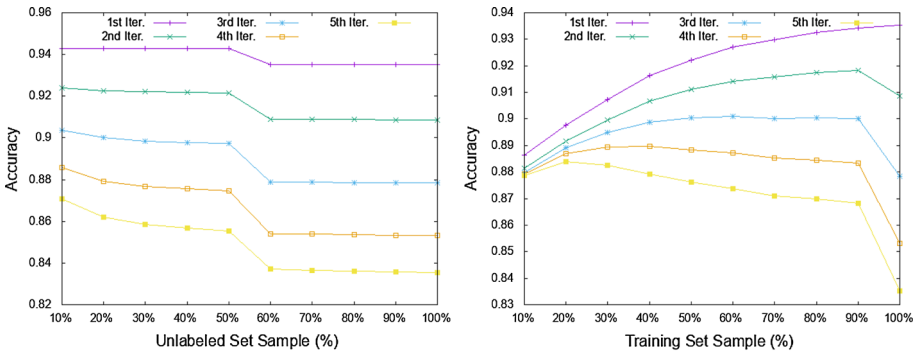
Fig. 8 Batch: effect of predicted instances used for dataset expansion (left) and effect of labeled set (right) using Co-Training and Self-Learning, per iteration

in Fig. 8a. As we can see, the accuracy drops as we further expand the dataset, for both Co-Training and Self-Learning. As the ratio of annotated instances increases, Co-Training (bigram's model) experiences a faster drop in its performance.

Same behavior is exhibited by EM, in Fig. 9a. As the unlabeled amount of instances is increasing, we observe that the accuracy is declining, faster for iterations four and five. When the unlabeled set reaches 50% of its original volume size, then EM algorithm starts making more and more errors which indicates noise in the unlabeled data. Note that such a drop is observed also when adding 10% of unlabeled data; at 10% however, the unlabeled data are still around 10 times larger than the labeled ones.

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 Fig. 8a). The results are depicted in Fig. 8b. 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.

The results of EM for the same experiment are depicted in Fig. 9b. 10% of the unlabeled data are employed by EM to find the maximum likelihood. One interesting observation is that the first three iterations are improving and the training set expands, while in the last two



(a) EM: Unlabeled set expansion **(b)** EM: Labeled set expansion

Fig. 9 Batch: effect of predicted instances used for dataset expansion (left) and effect of labeled set (right) using EM, per iteration

iterations accuracy declines. When the whole labeled set is used, only the first iteration is improving while all the rest are decreasing.

Due to the bad performance of EM in the batch case and for efficiency reasons (EM was significantly slower than the other two methods), we report hereafter only on Self-Learning and Co-Training. We plan to further investigate the EM performance in future research.

6.2 Performance of stream annotation

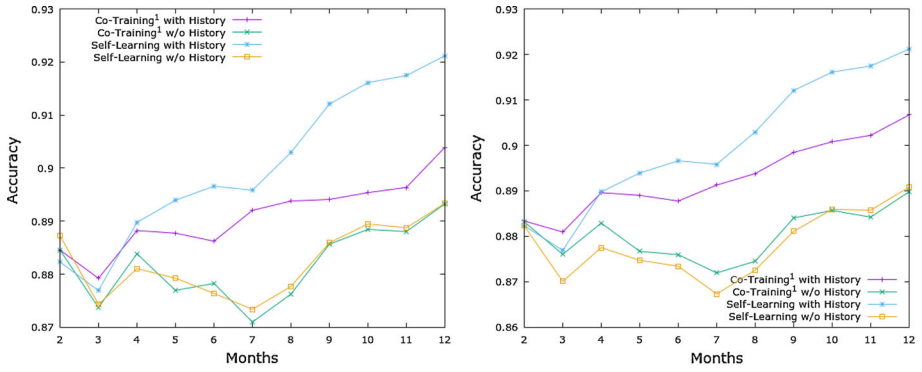
For the stream approach, we process the data on a monthly basis, and we evaluate how the temporal processing affects our methods. Let L_i be the labeled data (ground truth) for month i and let U_i be the corresponding unlabeled set. Our complete dataset therefore is a sequence of the form: $((L_1, U_1), (L_2, U_2), \ldots (L_{12}, U_{12}))$ 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 time point (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 spanning the whole period. 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 of the given month(s). That is, not all labeled data are used for training/testing, rather a sample of them according to the initial split.



(a) Stream: Holdout **(b)** Stream: Prequential

Fig. 10 Stream comparison between Co-Training and Self-Learning

6.2.1 Self-Learning versus Co-Training

The holdout evaluation is depicted in Fig. 10a, the prequential in Fig. 10b; 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 the best performance. So Co-Training is more effective with fewer 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; in particular, 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 i.e., month 4. The results are depicted in Fig. 11. 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 lighter models and therefore better efficiency (time, memory). The amount of data for each approach is depicted in Fig. 12: Labeled set is the original labeled data, and 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 Fig. 13, 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 toward 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.

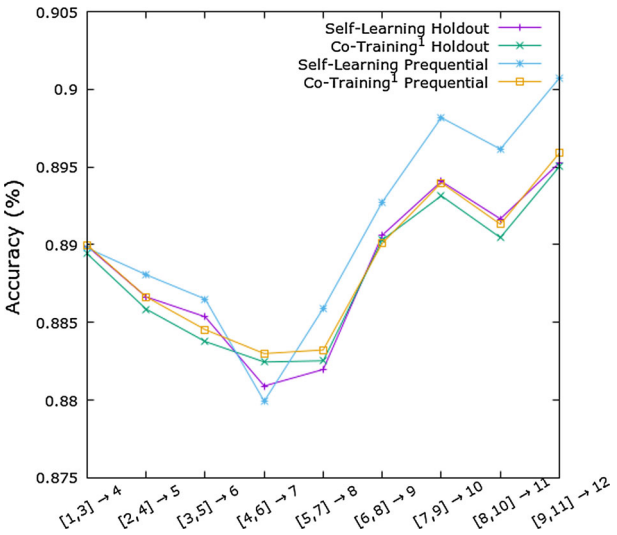


Fig. 11 Stream: sliding (3 months)

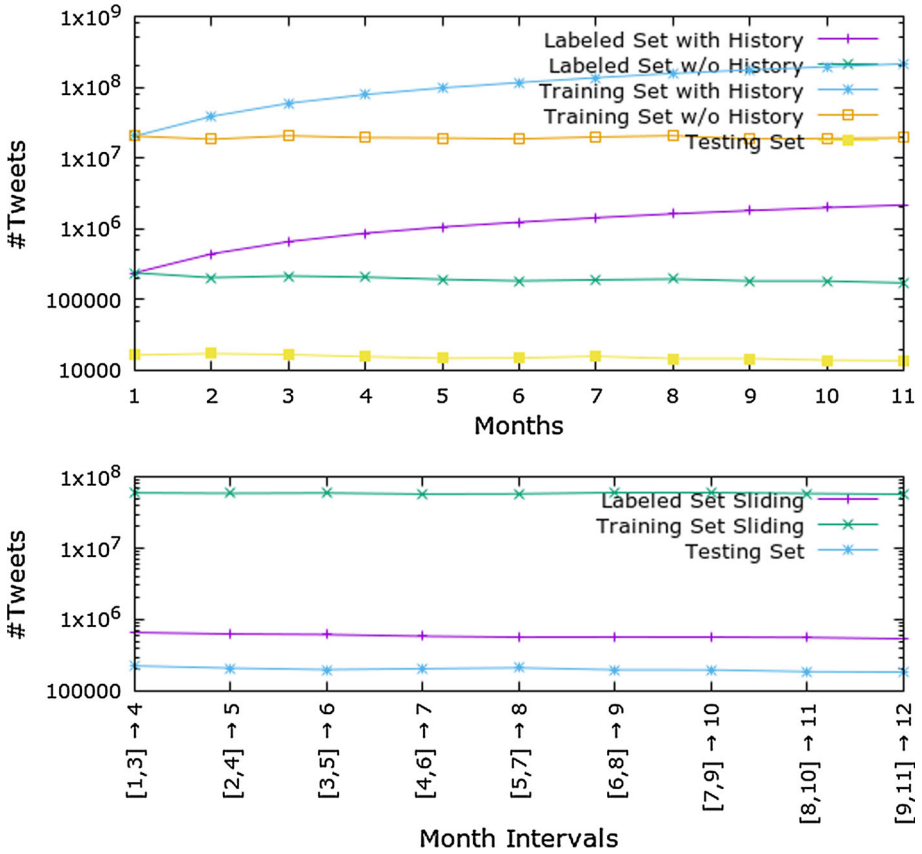


Fig. 12 Stream: prequential evaluation—cardinality of labeled, training, testing sets

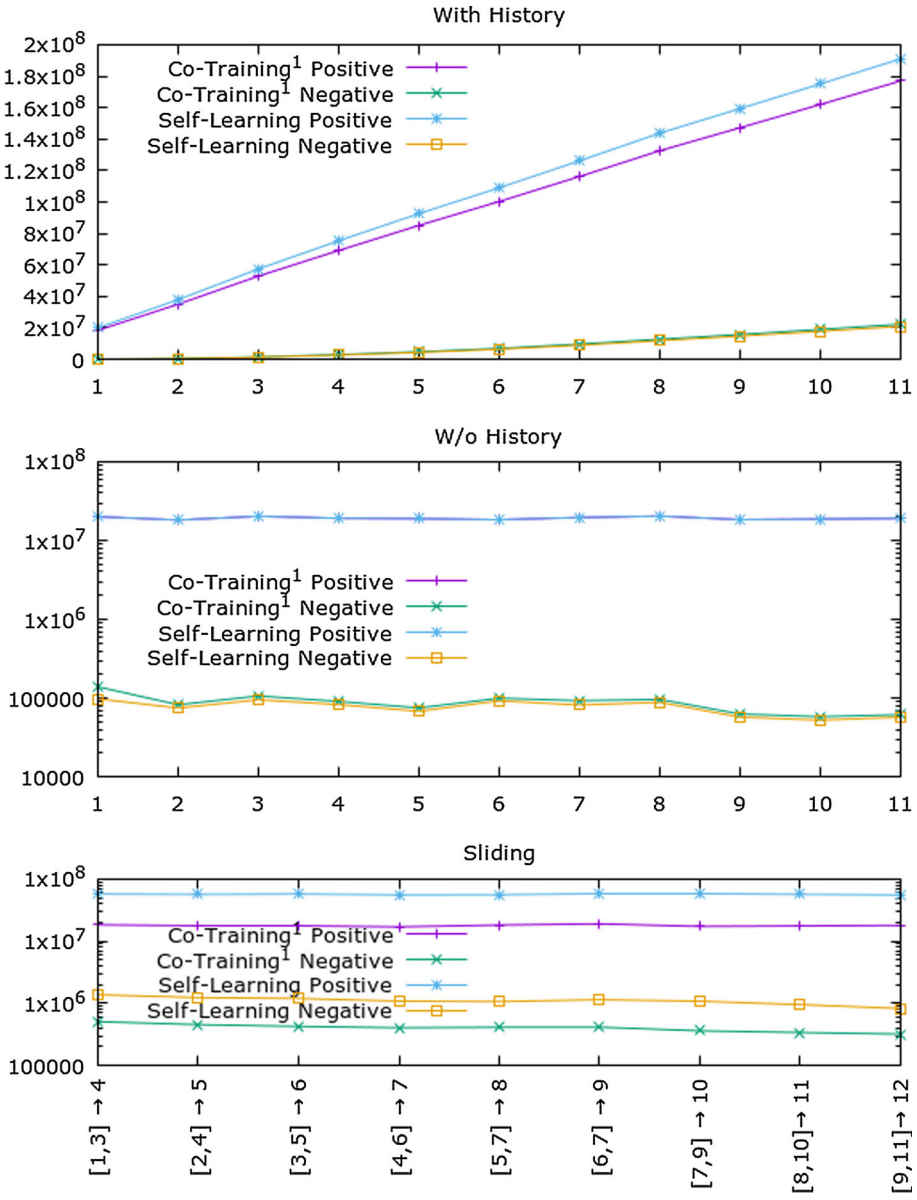


Fig. 13 Stream: class distribution of the annotated tweets over time

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, it results in more positive predictions.

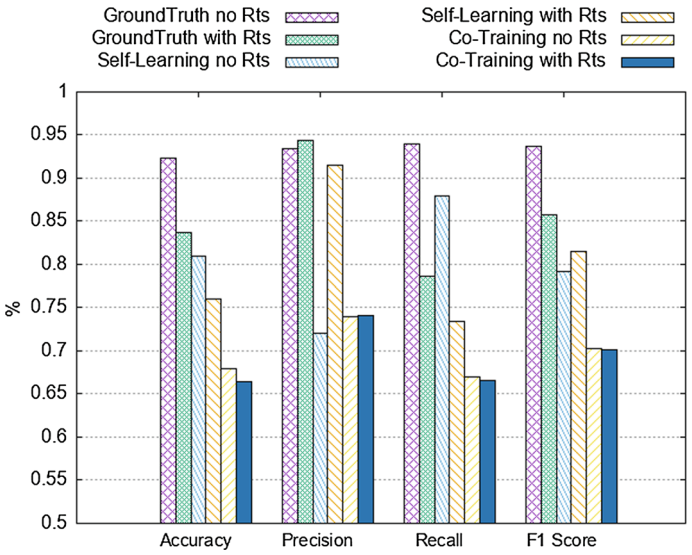


Fig. 14 Evaluation using Crowdflower platform

6.3 Crowd-sourcing evaluation

Except for the quantitative evaluation, we have also performed a qualitative evaluation of the derived labels by crowdsourcing. There is a plethora of crowd-sourcing platforms such as MTurk, CrowdFlower, CloudCrowd, ShortTask and MicroWorkers [59]. We chose CrowdFlower,[11] a platform in which annotators work on tasks such as data research tasks, transcription, categorization, and text production for product descriptions. Almost 5 million contributors have completed more than 1 billion tasks so far.

In total, 6000 tweets were annotated by the crowd, selected as follows: We randomly extracted 1000 tweets from each corpus: GroundTruth with retweets, GroundTruth without retweets, Self-Learning with retweets, Self-Learning without retweets, Co-Training with retweets and Co-Training without retweets. We requested three votes per tweet and filtered tweets which their average confidence score was less than 80%. The average label confidence is computed based on the confidence score given by each worker (in the range of [0, 100]). After the filtering, we ended up with 3928 crowd annotated tweets. Figure 14 depicts the evaluation over our datasets.

Datasets which do not contain retweets have better accuracy than datasets with retweets. We make these human-annotated tweets (3928) available to the community by providing the tweet id and the human-annotated label.

6.4 Comparing Self-Learning and Co-Training to SentiStrength and TreeBank

In this section, we compare our predictions to those of SentiStrength and TreeBank.

In Table 9, we compare SentiStrength to Self-Learning for each corpus: with retweets and without retweets. We see that for the corpus which does not contain retweets the negative percentage agreement is higher while the positive percentage agreement is lower in contrast

[11] http://www.crowdflower.com/.

Table 9 SentiStrength versus Self-Learning

Senti Stren.	No retweets Self-Learning		Retweets Self-Learning	
	Positive (%)	Negative (%)	Positive (%)	Negative (%)
Positive	**37.84**	15.25	**43.72**	16.47
Negative	11.00	**48.96**	8.00	**31.69**
Neutral	51.16	35.79	48.29	51.85

Agreements among predictions marked in boldface

Table 10 SentiStrength versus Co-Training

Senti Stren.	Unigrams				Bigrams			
	No retweets Co-Training		Retweets Co-Training		No retweets Co-Training		Retweets Co-Training	
	Positive (%)	Negative (%)	Positive (%)	Negative (%)	Positive (%)	Negative (%)	Positive (%)	Negative (%)
Positive	**39.37**	21.04	**42.30**	21.72	**38.09**	13.62	**44.37**	16.47
Negative	10.93	**30.62**	9.86	**25.79**	10.91	**48.89**	7.78	**31.10**
Neutral	49.70	48.35	47.84	52.49	51.00	37.49	47.85	52.43

Agreements among predictions marked in boldface

Table 11 TreeBank versus Self-Learning

TreeBank	No retweets Self-Learning		Retweets Self-Learning	
	Positive (%)	Negative (%)	Positive (%)	Negative (%)
Positive	**15.95**	4.40	**18.32**	4.60
Negative	40.50	**75.95**	37.42	**60.64**
Neutral	43.55	19.66	44.26	34.76

Agreements among predictions marked in boldface

Table 12 TreeBank versus Co-Training

TreeBank	Unigrams				Bigrams			
	No retweets Co-Training		Retweets Co-Training		No retweets Co-Training		Retweets Co-Training	
	Positive (%)	Negative (%)	Positive (%)	Negative (%)	Positive (%)	Negative (%)	Positive (%)	Negative (%)
Positive	**16.37**	6.10	**18.00**	6.73	**15.41**	3.64	**18.66**	4.55
Negative	38.77	**62.41**	37.33	**57.44**	41.51	**65.41**	36.95	**61.07**
Neutral	44.86	31.49	44.67	35.83	43.07	30.95	44.39	34.38

Agreements among predictions marked in boldface

Table 13 Batch: Self-Learning versus Co-Training, average accuracy for different δ (Retweets version)

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

to the corpus which contains retweets. The same behavior is observed in Table 10, where we compare SentiStrength to Co-Training. In both comparisons and in contrast to TreeBank (Self-Learning and Co-Training versus SentiStrength), we observe that SentiStrength tends to classify more tweets as neutral. This occurs due to our SentiStrength setup, in which we consider as neutral instances tweets that have same positive and negative score (absolute values). In many cases, positive and negative scores are equally high; however, SentiStrength cannot determine which sentiment is stronger than the other.

For Sentiment TreeBank, we observe the same behavior as in SentiStrength comparisons, where positive agreement increases and negative agreement decreases between the corpus without retweets and corpus with retweets (Tables 11, 12). Negative agreement in both comparisons is much higher than positive agreement. Moreover, we observe that the positive agreement, in both Self-Learning and Co-Training comparisons, is much lower than SentiStrength while the negative agreement is higher.

For the positively annotated tweets (by Self-Learning or Co-Training), TreeBank classifies most of them as negative compared to SentiStrength e.g., SentiStrength compared to Self-Learning has 11% disagreement, in the corpus without retweets, while TreeBank has 40% disagreement.

In conclusion, comparing the performance of methods trained upon different datasets is not easy, as already discussed in Sect. 3.4 regarding the ground truth comparison. Tweets are far different from comments or reviews due to the character limitations, and therefore, the structure of a tweet can vary significantly [28] compared to other texts. Moreover, the different spanning periods may lead to feature drifts, i.e., changes in the features/words or their relevance to the different classes [21,35].

6.5 Performance under redundancy (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. We perform holdout evaluation (67% training and 33% testing split) and report here on the most interesting findings. In Table 13, same setup as Table 8, 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 14, where we display the negative: positive class ratio for different δ values, for Self-

Table 14 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

Learning and Co-Training for the dataset with retweets and the dataset without retweets. It is clear that the class imbalance is significantly 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). In the retweets version, on the other hand, the redundancy acted to some extent as oversampling (though in our case, retweets come from both classes) and this helped the classifier to make more balanced predictions.

6.6 Performance of augmentation techniques

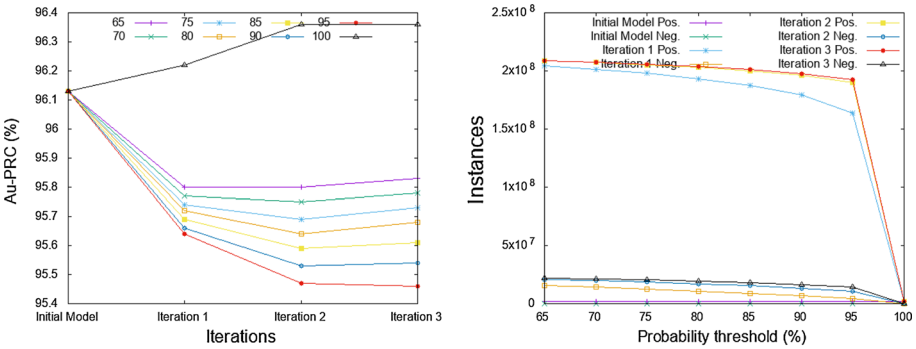
The performance of Co-Training and Self-Learning is evaluated using the area under precision–recall curve (AU-PRC) which better reflects classifier's performance, comparing to e.g., accuracy, in case of class imbalance [50] and also provides more informative representations than AUC [22]. We perform holdout evaluation (67% training and 33% testing split) and report AU-PRC and the number of predicted instances per iteration. In addition, we show the ratio of the annotated corpus (including ground truth while in Table 14 we show only the ratio of final predictions).

6.6.1 Self-Learning-based augmented batch annotation

Figure 15 demonstrates the performance of the original unmodified data same as in Sect. 6.1.1. We observe the same behavior as before, e.g., performance is degrading while δ is increased; however, for $\delta = 100\%$ performance is maximized while the labeled instances are significantly reduced compared to other thresholds.

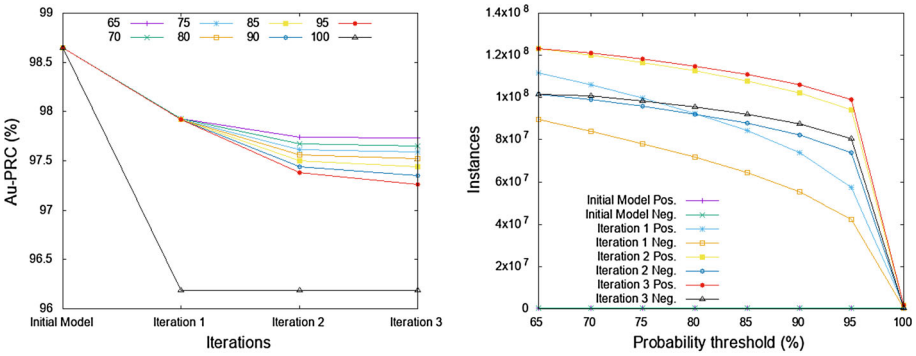
On the other hand, undersampling, oversampling, their combination and blankout methods have better performance for low δ values (Figs. 16, 17, 18 and 19). Moreover, by comparing these methods to the original Self-Learning procedure in Fig. 15, we observe that the performance is better when augmentation is employed. Also, the class imbalance problem is tackled as in each iteration the minority (negative) class receives more instances compared to the original Self-Learning.

However, word embeddings do not perform equally good as the other augmentation methods. In Fig. 20, we see that word embeddings have same behavior as original Self-Learning. Also, the negative instances are fewer in all the other methods. A probable reason of this



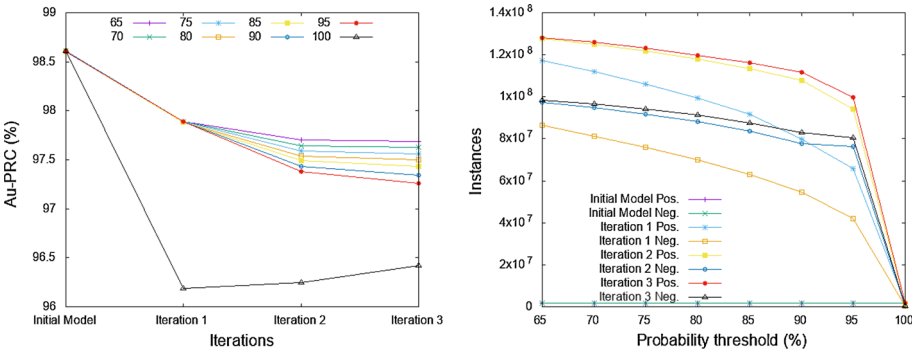
(a) Au-PRC **(b)** Number of instances per iteration

Fig. 15 Self-Learning original



(a) Au-PRC **(b)** Number of instances per iteration

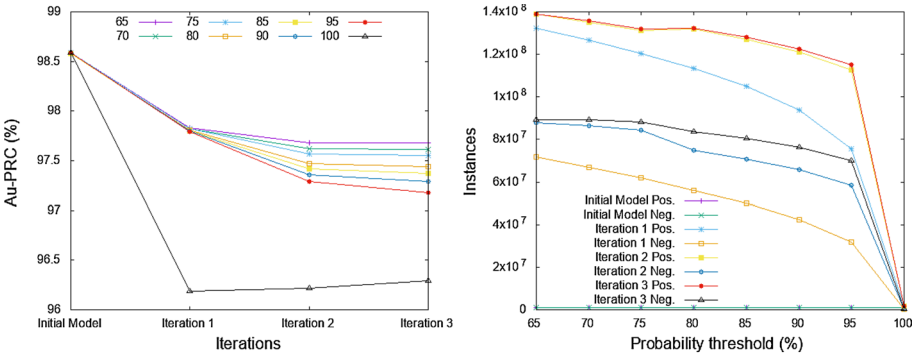
Fig. 16 Self-Learning combined with undersampling



(a) Au-PRC **(b)** Number of instances per iteration

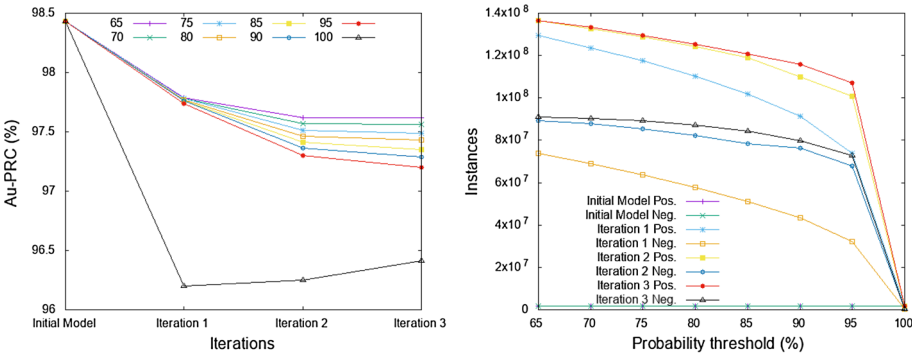
Fig. 17 Self-Learning combined with oversampling

behavior could be that the augmentation process generates pseudo-instances of the opposite class. Swapping terms with other semantically similar terms does not guarantee that the sentimentality of the terms will be the same. Even though we employ SentiWordNet to tackle



(a) Au-PRC **(b)** Number of instances per iteration

Fig. 18 Self-Learning combined with undersampling and oversampling



(a) Au-PRC **(b)** Number of instances per iteration

Fig. 19 Self-Learning combined with blankout

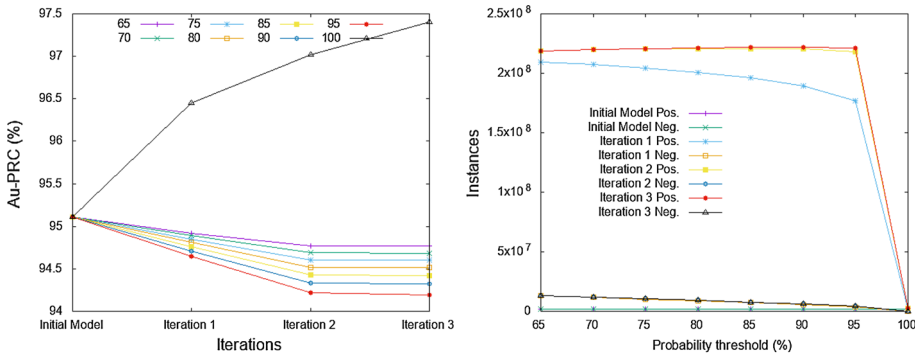
this problem, it is not enough due to the grammatical and syntactical structure of a tweet; therefore, the performance is degrading.

In Table 15, we show the ratio of negative, positive predicted instances per method. In contrast to Table 14, this table includes the ground truth while the latter shows the overall predicted instances, thus when $\delta = 100\%$ the difference is significantly reduced. Blankout, oversampling (Over.), undersampling (Under.) and the combination of the last two (Over. and Under.) methods tackle the problem of class imbalance. Word embeddings, on the other hand, are enhancing the gap between the two classes.

Furthermore, we have performed two significant tests: paired t test and McNemar's test, to compare original Self-Learning with the other augmentation methods. For every method, we obtained highly significant results in both tests (for $\tau = 0.01$).

6.6.2 Co-Training-based augmented batch annotation

For Co-Training, we report on both models: bigrams and unigrams in Figs. 21, 22, 23, 24, 25 and 26. In Fig. 21, the original Co-Training method is shown for which unigrams are slightly better than bigrams. Nonetheless, by comparing the class labels and performance of original

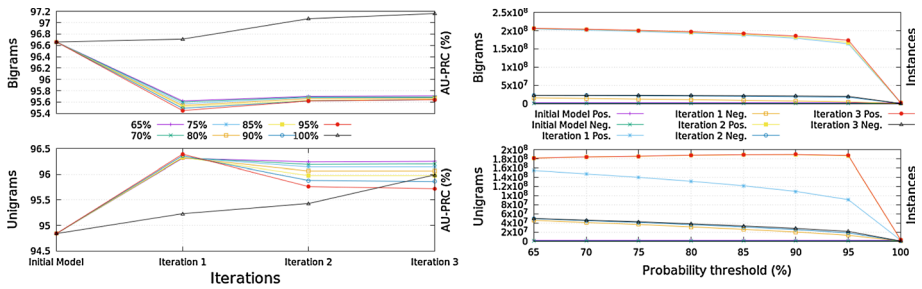


(a) Au-PRC **(b)** Number of instances per iteration

Fig. 20 Self-Learning combined with word embeddings

Table 15 Self-Learning: class ratio (negative:positive) of the predictions for different methods

δ (%)	Original	Blankout	Glove	Over.	Under.	Over. and Under.
65	1:9	1:2	1:16	1:1	1:1	1:2
70	1:9	1:1	1:18	1:1	1:1	1:2
75	1:9	1:1	1:21	1:1	1:1	1:2
80	1:9	1:1	1:24	1:1	1:1	1:2
85	1:9	1:1	1:29	1:1	1:1	1:2
90	1:10	1:1	1:36	1:1	1:1	1:2
95	1:11	1:1	1:52	1:1	1:1	1:2
100	1:10	1:7	1:11	1:7	1:7	1:7

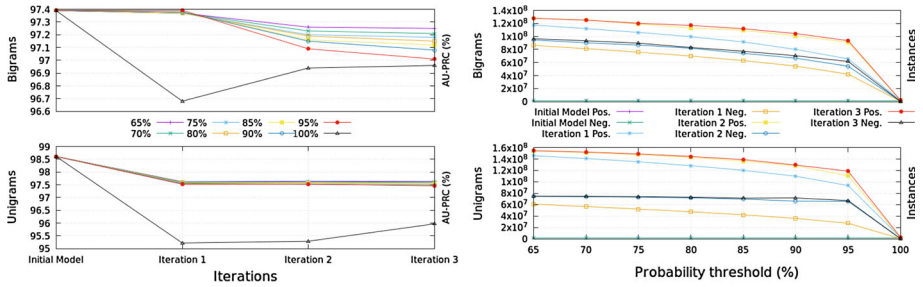


(a) Au-PRC **(b)** Number of instances per iteration

Fig. 21 Original Co-Training

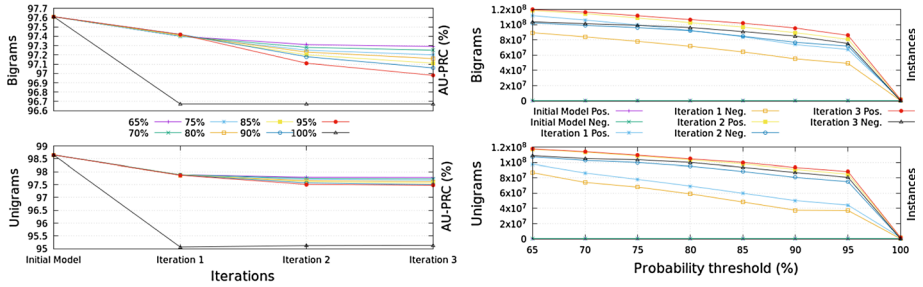
Co-Training with the augmented methods, we observe significant differences. However, same as in Self-Learning the word embeddings do not exhibit good performance compared to the other augmentation methods.

Nonetheless, by comparing Tables 16 and 17, we see that word embeddings reduce the gap between the two classes in the unigram model while the opposite is happening in the bigrams model. The latter model, however, is trained based on the predictions of the first model which implies that unigrams as feature space are affected more than bigrams. Other augmentation



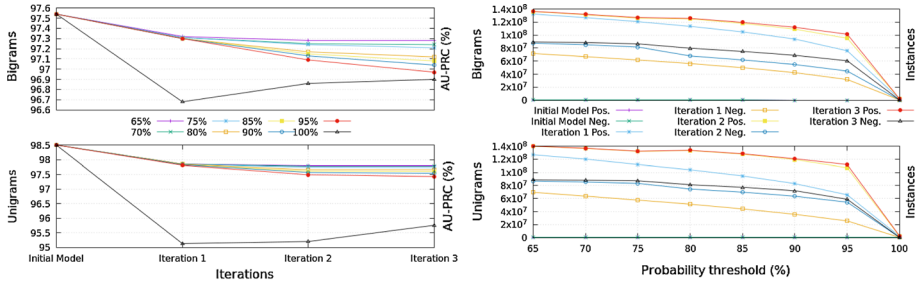
(a) Au-PRC **(b)** Number of instances per iteration

Fig. 22 Co-Training combined with oversampling



(a) Au-PRC **(b)** Number of instances per iteration

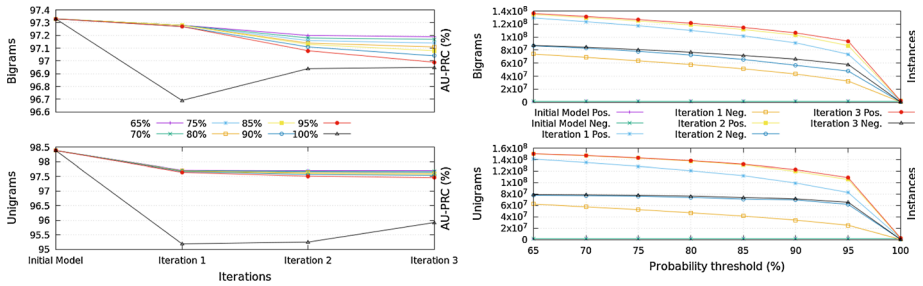
Fig. 23 Co-Training combined with undersampling



(a) Au-PRC **(b)** Number of instances per iteration

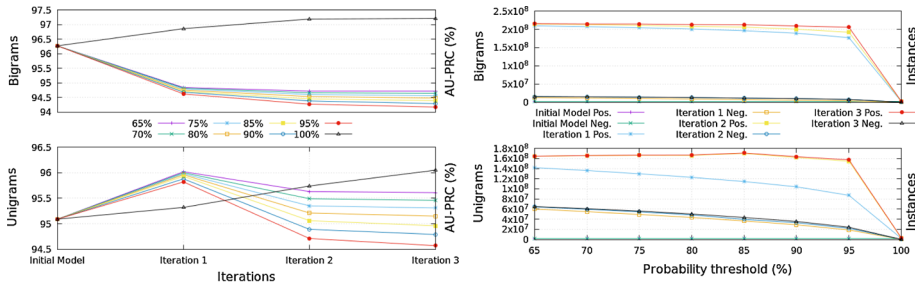
Fig. 24 Co-Training combined with over- and undersampling

methods such as oversampling and undersampling deal with class imbalance efficiently by balancing the two classes while maintaining high performance. In addition, same as in Self-Learning we performed the two significant tests, namely McNemar and paired t test, for which we compared the augmentation methods with the original Co-Training. For all the methods and the δ values, we obtained highly significant results (for $\tau = 0.01$).



(a) Au-PRC **(b)** Number of instances per iteration

Fig. 25 Co-Training combined with blankout



(a) Au-PRC **(b)** Number of instances per iteration

Fig. 26 Co-Training combined with embeddings

Table 16 Co-Training unigrams: class ratio (negative:positive) of the predictions for different methods

δ (%)	Original	Blankout	Glove	Over.	Under.	Over. and Under.
65	1:4	1:2	1:3	1:2	1:1	1:1
70	1:4	1:2	1:3	1:2	1:1	1:1
75	1:4	1:2	1:3	1:2	1:1	1:2
80	1:5	1:2	1:3	1:2	1:1	1:2
85	1:6	1:2	1:4	1:2	1:1	1:2
90	1:7	1:2	1:5	1:2	1:1	1:2
95	1:9	1:2	1:6	1:2	1:1	1:2
100	1:12	1:10	1:13	1:10	1:7	1:10

7 Conclusions

We presented how to annotate large textual collections with sentiment labels using distant supervision and semi-supervised learning. Our case study is TSentiment15, a 228 million tweets dataset with no retweets and 275 million tweets with retweets, spanning the whole year 2015. 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 [54].

Except for the annotated datasets (with and without retweets) which we make available to the community, our analysis resulted in interesting insights:

Table 17 Co-Training bigrams: class ratio (negative:positive) of the predictions for different methods

δ (%)	Original	Blankout	Glove	Over.	Under.	Over. and Under.
65	1:8	1:1	1:14	1:1	1:1	1:1
70	1:9	1:2	1:14	1:1	1:1	1:1
75	1:9	1:2	1:15	1:1	1:1	1:1
80	1:9	1:2	1:16	1:1	1:1	1:2
85	1:9	1:2	1:19	1:1	1:1	1:2
90	1:9	1:2	1:21	1:1	1:1	1:2
95	1:9	1:2	1:26	1:2	1:1	1:2
100	1:9	1:8	1:11	1:8	1:7	1:8

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., Fig. 8b) Self-Learning improves faster than Co-Training. Both approaches result in more positive predictions (c.f., Tables 4, 6), thus favoring the majority class. Self-Learning moreover propagates the original class imbalance to the successive iterations (c.f., Table 4). This is not the case for Co-Training (c.f., Table 6). Surprisingly, the performance of EM does not improve over the iterations, probably due to huge volume of unlabeled data ($|U|$ is almost 91 times larger than $|L|$) affecting the learner. A possible solution would be to select only instances that are labeled with a high probability for some class for the expansion; we leave this as part of future research. However, the predictions of the EM algorithm over the iterations became more balanced.

For streaming, (full) history helps with the performance (c.f., Fig. 10a, b). However, comparing to a sliding window approach, a sliding window of three months history (c.f., Fig. 11) performs almost equally well, while employing fewer data, thus offering a good trade-off. The batch approach is better than streaming in terms of accuracy; however, the latter is much more efficient.

In our learning setup we deal, except for the label scarcity problem, also with class imbalance with the positive sentiment class being highly overrepresented comparing to the negative class. To tackle the imbalance, we exploited data augmentation, namely semantic augmentation through word embeddings and corruption, as well as traditional oversampling and undersampling techniques. The goal of the augmentation process was to create more training data out of the existing training data by adding variation through domain meaningful and sound transformation. In both cases, it was our intention to preserve the class labels while keeping the tweets plausible; of course, this is not guaranteed as discussed already in the text, and therefore, augmentation might cause further degradation of the data quality. Based on the experiments in Sect. 6.6, we see that augmentation techniques tackle the problem of class imbalance while maintaining very high performance. Compared to original Self-Learning and Co-Training, methods achieved highly significant difference when equipped with augmentation methods (according to paired test and McNemar test).

In the augmentation direction, we also investigated the impact of dataset redundancy on performance. In particular, Twitter is characterized by high redundancy through retweets. Having such a redundant training set helped class imbalance as, to some extent, it resembles oversampling (though in our case, retweets come from both classes). By comparing the two versions (c.f., Tables 8, 13), we observe that the retweet version has lower performance compared to the version which does not contain retweets. However, there is a huge difference

w.r.t. class imbalance between these two versions which can be observed in Table 14. We carefully removed duplicates from the test set to ensure that our evaluation is not affected by the redundancy (a similar effect has been studied for recommendation systems in Basaran et al. [4]).

In our future work, we will investigate further data augmentation techniques as a tool for expanding a training set in meaningful ways and tackling problems like class imbalance. In particular, we want to focus on how to identify meaningful augmentations for a given domain and also on what parts of the population should be augmented to avoid noise generation and degradation of data quality.

Moreover, augmentation typically refers to label-preserving transformations; however, in certain applications including text, instances of the "opposite" class could be generated. In text, for example one can turn a positive text into a negative one using negations e.g., like "I like summer" → "I dont like summer" or "clever" → "stupid." Finally, we will investigate the impact of refined word embeddings w.r.t. the sentiment task [64] for our semantic augmentation procedure.

Acknowledgements The work was inspired by the German Research Foundation (DFG) Project (Grant No. 317686254) OSCAR (Opinion Stream Classification with Ensembles and Active leaRners) for which the last author is a Principal Investigator.

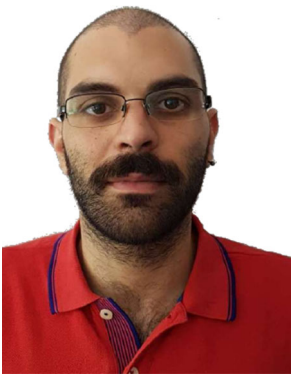
References

1. Aue A, Gamon M (2005) Customizing sentiment classifiers to new domains: a case study. In: Proceedings of recent advances in natural language processing (RANLP), vol 1, pp 2–1
2. Baccianella S, Esuli A, Sebastiani F (2010) Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In: LREC, vol 10, pp 2200–2204
3. Balcan M-F, Blum A, Yang K (2005) Co-training and expansion: towards bridging theory and practice. In: Advances in neural information processing systems, pp 89–96
4. Basaran D, Ntoutsi E, Zimek A (2017) Redundancies in data and their effect on the evaluation of recommendation systems: a case study on the amazon reviews datasets. In: Proceedings of the 2017 SIAM international conference on data mining, pp 390–398. SIAM
5. Berardi G, Esuli A, Sebastiani F, Silvestri F (2013) Endorsements and rebuttals in blog distillation. Inf Sci 249:38–47
6. Bifet A, Frank E (2010) Sentiment knowledge discovery in twitter streaming data. In: International conference on discovery science. Springer, Berlin, pp 1–15
7. Biyani P, Caragea C, Mitra P, Zhou C, Yen J, Greer GE, Portier K (2013) Co-training over domain-independent and domain-dependent features for sentiment analysis of an online cancer support community. In: 2013 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM), pp 413–417. IEEE
8. Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on computational learning theory, pp 92–100. ACM
9. Cozman FG, Cohen I, Cirelo MC (2003) Semi-supervised learning of mixture models. In: Proceedings of the 20th international conference on machine learning (ICML-03), pp 99–106
10. Dasgupta S, Ng V (2009) 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, vol 2, pp 701–709. Association for Computational Linguistics
11. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B (Methodol) 39:1–22
12. Drummond C, Holte RC et al (2003) C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In: Workshop on learning from imbalanced datasets II, vol 11, pp 1–8. Citeseer
13. Du J, Ling CX, Zhou Z-H (2011) When does cotraining work in real data? IEEE Trans Knowl Data Eng 23(5):788–799

14. Estabrooks A, Jo T, Japkowicz N (2004) A multiple resampling method for learning from imbalanced data sets. Comput Intell 20(1):18–36
15. Fafalios P, Iosifidis V, Ntoutsi E, Dietze S (2018a) Tweetskb: a public and large-scale RDF corpus of annotated tweets. In: European semantic web conference. Springer, Berlin, pp 177–190
16. Fafalios P, Iosifidis V, Stefanidis K, Ntoutsi E (2018b) Tracking the history and evolution of entities: entity-centric temporal analysis of large social media archives. Int J Digit Lib 1–13. https://doi.org/10.1007/s00799-018-0257-7
17. Fralick S (1967) Learning to recognize patterns without a teacher. IEEE Trans Inf Theory 13(1):57–64
18. Gatti L, Guerini M, Turchi M (2016) Sentiwords: deriving a high precision and high coverage lexicon for sentiment analysis. IEEE Trans Affect Comput 7(4):409–421
19. Globerson A, Roweis S (2006) Nightmare at test time: robust learning by feature deletion. In: Proceedings of the 23rd international conference on machine learning, pp 353–360. ACM
20. Go A, Bhayani R, Huang L (2009) Twitter sentiment classification using distant supervision. CS224N Proj Rep Stanf 1(12):2009
21. Hamilton WL, Leskovec J, Jurafsky D (2016) Diachronic word embeddings reveal statistical laws of semantic change. arXiv preprint arXiv:1605.09096
22. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 19:1263–1284
23. He H, Ma Y (2013) Imbalanced learning: foundations, algorithms, and applications. Wiley, New York
24. He Y, Zhou D (2011) Self-training from labeled features for sentiment analysis. Inf Process Manag 47(4):606–616
25. Hube C, Fetahu B (2019) Neural based statement classification for biased language. In: Proceedings of the twelfth ACM international conference on web search and data mining, pp 195–203. ACM
26. Iosifidis V, Ntoutsi E (2017) Large scale sentiment learning with limited labels. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1823–1832. ACM
27. Iosifidis V, Oelschlager A, Ntoutsi E (2017) Sentiment classification over opinionated data streams through informed model adaptation. In: International conference on theory and practice of digital libraries, pp 369–381. Springer, Berlin
28. Kaufmann M, Kalita J (2010) Syntactic normalization of twitter messages. In: International conference on natural language processing, Kharagpur, India
29. Kucuktunc O, Cambazoglu BB, Weber I, Ferhatosmanoglu H (2012) A large-scale sentiment analysis for yahoo! answers. In: Proceedings of the fifth ACM international conference on Web search and data mining, pp 633–642. ACM
30. Li S, Wang Z, Zhou G, Lee SYM (2011) Semi-supervised learning for imbalanced sentiment classification. In: IJCAI proceedings-international joint conference on artificial intelligence, vol 22, pp 1826
31. Liu S, Zhu W, Xu N, Li F, Cheng X-q, Liu Y, Wang Y (2013a) Co-training and visualizing sentiment evolvement for tweet events. In: Proceedings of the 22nd international conference on World Wide Web, pp 105–106. ACM
32. Liu Y, Yu X, An A, Huang X (2013b) Riding the tide of sentiment change: sentiment analysis with evolving online reviews. World Wide Web 16(4):477–496 ISSN 1386-145X
33. Lucas M, Downey D (2013) Scaling semi-supervised naive bayes with feature marginals. In: Proceedings of the 51st annual meeting of the association for computational linguistics (Volume 1: Long Papers), vol 1, pp 343–351
34. Melidis DP, Campero AV, Iosifidis V, Ntoutsi E, Spiliopoulou M (2018a) Enriching lexicons with ephemeral words for sentiment analysis in social streams. In: Proceedings of the 8th international conference on web intelligence, mining and semantics, p 38. ACM
35. Melidis DP, Spiliopoulou M, Ntoutsi E (2018b) Learning under feature drifts in textual streams. In: Proceedings of the 27th ACM international conference on information and knowledge management, CIKM '18, pp 527–536, New York, USA. ACM. ISBN 978-1-4503-6014-2
36. Melville P, Gryc W, Lawrence RD (2009) Sentiment analysis of blogs by combining lexical knowledge with text classification. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 1275–1284. ACM
37. Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, Freeman J, Tsai D, Amde M, Owen S et al (2016) Mllib: Machine learning in apache spark. J Mach Learn Res 17(34):1–7
38. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
39. Mohammad SM, Kiritchenko S, Zhu X (2013) NRC-Canada: building the state-of-the-art in sentiment analysis of tweets. arXiv preprint arXiv:1308.6242
40. Nigam K, Ghani R (2000) Analyzing the effectiveness and applicability of co-training. In: Proceedings of the ninth international conference on Information and knowledge management, pp 86–93. ACM

41. Nigam K, McCallum AK, Thrun S, Mitchell T (2000) Text classification from labeled and unlabeled documents using em. Mach Learn 39(2–3):103–134
42. Nigam K, McCallum A, Mitchell T (2006) Semi-supervised text classification using EM. In: Chapelle O, Scholkopf B, Zien A (eds) Semi-supervised learning. MIT Press. https://doi.org/10.7551/mitpress/9780262033589.003.0003
43. Nigam KP (2001) Using unlabeled data to improve text classification. Technical report, Carnegie-mellon univ Pittsburgh pa school of computer science
44. Paltoglou G, Thelwall M (2010) A study of information retrieval weighting schemes for sentiment analysis. In: Proceedings of the 48th annual meeting of the association for computational linguistics, pp 1386–1395. Association for Computational Linguistics
45. Pan SJ, Ni X, Sun J-T, Yang Q, Chen Z (2010) Cross-domain sentiment classification via spectral feature alignment. In: Proceedings of the 19th international conference on World wide web, pp 751–760. ACM
46. Pang B, Lee L (2005) Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the 43rd annual meeting on association for computational linguistics, pp 115–124. Association for Computational Linguistics
47. Pang B, Lee L, Vaithyanathan S (2002) Thumbs up?: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing, vol 10, pp 79–86. Association for Computational Linguistics
48. Pang B, Lee L et al (2008) Opinion mining and sentiment analysis. Found Trends® Inf Retr 2(1–2):1–135
49. Pennington J, Socher R, Manning C (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
50. Saito T, Rehmsmeier M (2015) The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PloS One 10(3):e0118432
51. Sedhai S, Sun A (2015) Hspam14: a collection of 14 million tweets for hashtag-oriented spam research. In: SIGIR, pp 223–232. ACM
52. Silva NFFD, Coletta LF, Hruschka ER (2016) A survey and comparative study of tweet sentiment analysis via semi-supervised learning. ACM Comput Surv (CSUR) 49(1):15
53. Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng A, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 conference on empirical methods in natural language processing, pp 1631–1642
54. Spiliopoulou M, Ntoutsi E, Zimmermann M (2017) Opinion stream mining. In: Sammut C, Webb GI (eds) Encyclopedia of machine learning and data mining. Springer, Boston, MA
55. Su J, Shirab JS, Matwin S (2011) Large scale text classification using semi-supervised multinomial naive bayes. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp 97–104. Citeseer
56. Tapia PA, Velásquez JD (2014) Twitter sentiment polarity analysis: a novel approach for improving the automated labeling in a text corpora. In: International conference on active media technology, pp 274–285. Springer, Berlin
57. Toutanova K, Klein D, Manning CD, Singer Y (2003) 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, vol 1, pp 173–180. Association for Computational Linguistics
58. Unnikrishnan V, Beyer C, Matuszyk P, Niemann U, Pryss R, Schlee W, Ntoutsi E, Spiliopoulou M (2018) Entity-level stream classification: exploiting entity similarity to label the future observations referring to an entity. In: 2018 IEEE 5th international conference on data science and advanced analytics (DSAA), pp 246–255. IEEE
59. Vakharia D, Lease M (2013) Beyond AMT: an analysis of crowd work platforms. arXiv preprint arXiv:1310.1672
60. Wagner S, Zimmermann M, Ntoutsi E, Spiliopoulou M (2015) Ageing-based multinomial naive bayes classifiers over opinionated data streams. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, Berlin, pp 401–416
61. Wang S, Manning CD (2012) Baselines and bigrams: Simple, good sentiment and topic classification. In: Proceedings of the 50th annual meeting of the association for computational linguistics: short papers, vol 2, pp 90–94. Association for Computational Linguistics
62. Xia R, Wang C, Dai X-Y, Li T (2015) Co-training for semi-supervised sentiment classification based on dual-view bags-of-words representation. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (Volume 1: Long Papers), vol 1, pp 1054–1063
63. Ye Q, Zhang Z, Law R (2009) Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. Exp Syst Appl 36(3):6527–6535

64. Yu L-C, Wang J, Lai KR, Zhang X (2017) Refining word embeddings for sentiment analysis. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp 534–539
65. Zhang M, Tang J, Zhang X, Xue X (2014) Addressing cold start in recommender systems: a semi-supervised co-training algorithm. In: Proceedings of the 37th international ACM SIGIR conference on research and development in information retrieval, pp 73–82. ACM
66. Zhao L, Huang M, Yao Z, Su R, Jiang Y, Zhu X (2016) Semi-supervised multinomial naive bayes for text classification by leveraging word-level statistical constraint. In: Thirtieth AAAI conference on artificial intelligence
67. Zhu X, Goldberg AB, Brachman R, Dietterich T (2009) Introduction to semi-supervised learning. Morgan and Claypool Publishers, Los Altos ISBN 1598295470, 9781598295474
68. Zimmerann M, Ntoutsi E, Spiliopoulou M (2014) A semi-supervised self-adaptive classifier over opinionated streams. In: 2014 IEEE international conference on data mining workshop, pp 425–432. IEEE



Vasileios Iosifidis is a Ph.D. student at the Leibniz University of Hanover. He received his Diploma (2014) in Computer Engineering as well as his Master (2016) in Software Engineering from the University of Patras, Greece. Prior to starting his PhD, he was a researcher at Computer Technology Institute and Press "Diophantus." His research interests include data mining, stream learning, semi-supervised learning and fairness-aware learning.

Eirini Ntoutsi is an associate professor at the Faculty of Electrical Engineering and Computer Science at the Leibniz University Hanover (LUH) and member of the L3S Research Center. Her research lies in the areas of Data Mining and Machine Learning where she develops methods for learning over complex data and data streams as well as methods for fairness-aware machine learning. Prior to joining LUH, she was a post-doctoral researcher at Ludwig-Maximilians University, Munich. She obtained her PhD (2008) from the University of Piraeus, Athens and she holds an MSc (2003) and a Diploma (2001) in computer engineering both from the Department of Computer Engineering and Informatics, Patras, Greece.