Active Stream Learning with an Oracle of Unknown Availability for Sentiment Prediction

Elson Serrao and Myra Spiliopoulou

Otto-von-Guericke-University Magdeburg, Germany
elson.serrao@gmail.com, myra@ovgu.de

Abstract. Active learning holds the promise of learning models from the data with minimal expert input. However, it assumes that the expert is always available or only at the beginning. We waive this assumption and investigate to what extent active learning is effective in practice. We focus on sentiment classification over real streams of opinions. We show that at least for the two real streams we have analyzed, the random strategy is very competitive, and querying the expert in an intelligent way does not bring many advantages, at least when the expert is irregularly available.

Keywords: active learning, oracle availablity, polarity model learning, opinion stream mining

1 Introduction

The objective of active learning is to obtain better or comparable performance to a fully supervised learner with fewer labels if the learner is given the opportunity to select the instances for which it requires labels [1]. Active learning is thus very suitable in those scenarios where there is an abundance of unlabeled data and obtaining new labels is rather expensive. Labels are obtained using an oracle who can, for example, be a domain expert or a human annotator from a crowd-sourcing platform. However, it is often assumed that there is a single oracle that is always correct, always available and inexpensive to query. While there are surveys [1], [2], and [3] and studies [4], [5] that provide insights to the above mentioned challenges in active learning, only a few studies focus on the availability of the oracle for streaming data [6].

In this paper, we consider a stream of opinionated documents and try to predict the sentiment of the document as being either positive, negative or neutral. Over time drift may be observed in the stream due to evolving topics, data and vocabulary, requiring the classifier to adapt to the opinionated stream. For this we use active learning to obtain new labels from the data stream. Instances to be labeled by the oracle are sampled using an appropriate query strategy. However, we assume that the oracle is available irregularly i.e. according to a pattern unknown to the learner. This implies that the oracle may be queried at each moment, but will respond by delivering the label only if it is available.

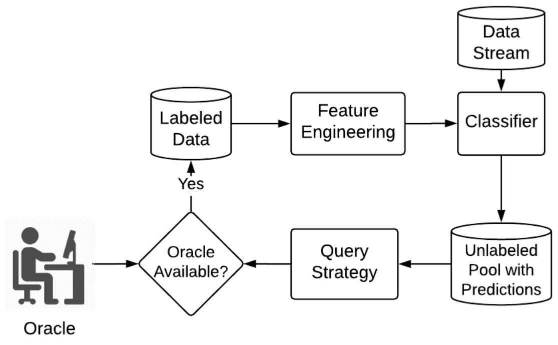


Fig. 1. Interaction of the stream learner and an irregularly available oracle

If the oracle is unavailable, the instance is not used. This workflow is shown in Figure 1.

The remaining of the paper is organized as follows. Section 2 discusses the related work. In Section 3 we detail our framework and the active learning query strategies used. Section 4 describes the setup for our experiments. While in Section 5 we discuss the results of those experiments. We present our concluding remarks in Section 6.

2 Related Work

Most of the algorithms for active learning on streams either assume infinite verification latency, whereupon they invoke semi-supervised learning [7] or they assume that the Oracle is always available to provide labels [8], [9], [10], [11], [12].

Shickel and Rashidi in [6] propose a framework that is aware of the oracle's availability for data streams. Their framework considers multiple oracles and focuses on querying first those oracles that have a higher availability. They try to achieve a cost-benefit tradeoff by using a dynamic labeling budget proportional to the oracle's availability with the cost of labeling an instance inversely proportional to the oracle's availability. However, such a cost-benefit tradeoff seems unrealistic in real-world scenarios where the cost is likely to be regulated by the difficulty in obtaining the label and other factors [5]. No experiments were conducted with oracles of varying expertise, which is often seen in active learning literature considering multiple oracles.

3 Active Learning on an Opinionated Stream

We consider a data stream $\mathcal{D}$ of opinionated documents observed at distinct timepoints $t_0, t_1, \ldots, t_i, \ldots$ where at each timepoint t_i we receive a batch of documents. We define the timepoint on a temporal level where, for example, the timepoint could be a week. Consequently, all the documents arriving during the time period from t_{i-1} to t_i would comprise the batch of documents for t_i.

Our framework encompasses an algorithmic core described in Section 3.1 and the query strategies in Section 3.2. We link this framework with a simulator of the oracle's availability, described in Section 4.1, and with an algorithm for the preparation of the opinionated data stream, described in Section 4.2.

3.1 Modeling Oracle Unavailability during Querying

We consider the beginning of the stream at t_0 to be characterized by the availability of an initial set of labeled documents L_0. The initially labeled documents L_0 are used to initialize the classifier $\varDelta$. At subsequent timepoints i.e. t_1 onwards, we receive unlabeled data U_t. If the budget $\mathcal{B}$ is not exceeded, for every unlabeled document x, we use the trained classifier $\varDelta$ to predict the probability $P(\hat{y}_c|x)\ \forall c \in C$, where c represents the sentiment of the document, namely, positive, negative or neutral. Our method calculates the confidence of the learner's prediction $\mathcal{I}$ using metric ϕ, and launches a request for the true label y when necessary. The oracle provides the true label y only if it is available and the document x is added to the labeled data for the next iteration. In the event that the oracle is unavailable, x is not used to adapt the learner. An overview of our framework is shown in Algorithm 1.

We assume the cost of labeling is the same for every document at any time t. If n_t is the number of queries sent to the oracle at t, then the utilized budget at t is given by

$$\frac{n_t}{|U_t|} < \mathcal{B} \tag{1}$$

To adapt to the evolving data stream, we utilize a sliding window $\mathcal{W}$ [13]. At every timepoint t, we add the labeled documents L_t to the window. Once the window is full, the documents from the oldest timepoint within the window are forgotten. Depending on the chosen classifier $\varDelta$, for every iteration, there may be a need to retrain $\varDelta$ with the documents in the window.

Although the framework is capable of using any confidence metric such as maximum posterior probability or the maximum margin between the first and second most probable class, we propose calculating the confidence of a prediction using entropy as,

$$\phi_H = 1 - \left[-\sum_{c \in C} P(\hat{y}_c|x) \log_{|C|} P(\hat{y}_c|x) \right] \tag{2}$$

3.2 Query Strategies

We use uncertainty based query strategies for the active learner. We also use the variable uncertainty and variable randomized uncertainty strategies introduced by Žliobaitė et al. in [9], [10] and [11]. The difference in our implementation is that we have generalized the strategies to allow the use of any confidence metric while determining if an instance needs to be sampled.

Random Strategy: This strategy randomly selects an instance to be labeled by the oracle with a probability given by the budget $\mathcal{B}$. In this sense, it is very naive and is used as the baseline strategy.

Algorithm 1: Active Learning with an Irregularly Available Oracle

```
Input: $\Delta$ - classifier with relevant parameters; $size$ - window size; $\mathcal{B}$ - budget;
       $o$ - oracle; $\phi$ - confidence metric; queryStrategy - query strategy with
       parameters;
Initialize: $t \leftarrow 0$; $\mathcal{W} \leftarrow$ SlidingWindow($size$)
1  Receive labeled data $L_t$
2  $\mathcal{W} \leftarrow$ addToWindow($L_t$)
3  for $t = 1, 2, \dots$ do
4      Train classifier $\Delta$ with instances in $\mathcal{W}$
5      Receive unlabeled data $U_t$
6      $n_t \leftarrow 0$
       $L_t \leftarrow \emptyset$
7      for each instance $x \in U_t$ do
8          $P_c \leftarrow P_\Delta(\hat{y}_c|x)\ \forall c \in C$ // predict the probability
9          $\hat{y} \leftarrow \arg\max_c(P_c)$ // predicted label
10         if $(n_t/|U_t|) < \mathcal{B}$ then
11             $\mathcal{I} \leftarrow \phi(P_c)$ // compute the confidence
12             if queryStrategy($\mathcal{B}, \mathcal{I}, \dots$) $= True$ then
13                 $n_t \leftarrow n_t + 1$
14                 if isAvailable($o$) then
15                     $y \leftarrow$ get true label of $x$ from the oracle $o$
                       $L_t \leftarrow L_t \cup (x, y)$
16     $\mathcal{W} \leftarrow$ addToWindow($L_t$)
```

Fixed Uncertainty Strategy: This strategy samples those instances which the learner are least confident of by comparing the confidence of prediction of an instance to a fixed threshold θ.

Variable Uncertainty Strategy: For a learner to adapt to an evolving data stream, obtaining labels for the least confident instances within each time-point would be more beneficial. The variable uncertainty strategy described in Algorithm 2 provides a variable threshold that adjusts itself to the incoming data stream. When the data stream speeds up, the confidence of the learner decreases. In such cases, it decreases its threshold so that least confident instances are queried first. On the other hand, at times when the learner is confident of its prediction, the threshold is increased to capture the most uncertain instances.

Algorithm 2: VariableUncertainty($\mathcal{I}$, s)

```
Input: $\mathcal{I}$ - confidence score; $s \in (0, 1]$ - threshold adjustment step
Output: $True$ if true label is required else $False$
Initialize: labeling threshold $\theta \leftarrow 1$
1  if $\mathcal{I} < \theta$ then
2      $\theta \leftarrow \theta(1 - s)$// uncertain instance: decrease the threshold
3      return $True$
4  else
5      $\theta \leftarrow \theta(1 + s)$// confident instance: increase the threshold
6      return $False$
```

Variable Randomized Uncertainty Strategy: Uncertainty based active learning query strategies generally focus on sampling those instances that are close to the decision boundary of the learner. In evolving data streams changes may occur anywhere in the instance space. So as to not miss the change that may occur elsewhere, the variable randomized uncertainty strategy occasionally samples those instances that the learner is confident of. As shown in Algorithm 3, the threshold is multiplied by a normally distributed random variable to sample the confident instances.

```
Algorithm 3: VariableRandomizedUncertainty($\mathcal{I}$, s, $\delta$)
  Input: $\mathcal{I}$ - confidence score; $s \in (0,1]$ - threshold adjustment step
         $\delta$ - variance of threshold randomization
  Output: $True$ if true label is required else $False$
  Initialize: labeling threshold $\theta \leftarrow 1$
1 $\theta_{randomized} \leftarrow \theta \times \eta$, where $\eta \in \mathcal{N}(1, \delta)$ is a random multiplier
2 if $\mathcal{I} < \theta_{randomized}$ then
3     $\theta \leftarrow \theta(1-s)$// uncertain instance: decrease the threshold
4     return $True$
5 else
6     $\theta \leftarrow \theta(1+s)$ // confident instance: increase the threshold
7     return $False$
```

4 Experiment Setup

The goal of our experiments is to study how the performance of the learner is affected when the availability of the oracle changes. The following sub-sections describe the oracle availability simulator, the datasets used, and the evaluation criteria and strategy.

4.1 Simulator of Oracle Availability

At timepoint t, we consider the oracle to be available with a probability of α_t. In our experiments, at any timepoint t, we set the oracle's availability $\alpha_t = \alpha$ and is considered to be independent of the availability at $t-1$. Algorithm 4 provides a more formal definition of our simulator.

```
Algorithm 4: Oracle Availability Simulator
  Input: $\alpha \in (0,1]$ - availability of the oracle;
  Output: $True$ if the oracle is available else $False$
1 return uniform(0, 1) $\leq \alpha$
```

4.2 Datasets and Feature Engineering

Yelp: The Yelp Dataset [1] contains about 5.2 million reviews of various businesses over a period of 13 years from 11 metropolitan areas across 4 countries. We

[1] https://www.yelp.com/dataset/challenge

filtered the dataset for English reviews and additionally removed those reviews whose length was less than 15 words. For our data stream, we considered the reviews from 2009 onwards.

Amazon: The Amazon dataset introduced in [14], contains reviews of several product categories. Using the 5-core datasets of the product categories, we build a dataset with an intention of introducing concept drift. For this purpose, we randomly selected three product categories every three months from among the following nine categories: *Home and Kitchen*, *Kindle Store*, *Health and Personal Care*, *Cell Phones and Accessories*, *Apps for Android*, *Electronics*, *Clothing, Shoes and Jewelry*, *CDs and Vinyl*, and *Beauty*. We removed the duplicate reviews arising from the product having multiple categories.

Both the Yelp and Amazon employ a 5-star rating scheme, where 5-stars is the highest rating while 1-star is the lowest. We considered the 1 and 2-star rating to be negative, 3-star rating neutral and 4 and 5-star rating positive.

Feature Engineering: We preprocessed the reviews by replacing URLs, negations and currencies with the tokens URL, NEGATION and CURRENCY respectively and some emoticons with tokens like SMILE and HEART. We further suppressed repeated letters, expanded contractions (e.g. "I'm" into "I am"), removed stopwords and replaced words by their lemmas.

After preprocessing, we extracted features from the reviews using word n-grams with $n = 3$ along with its corresponding frequency of occurrence. We selected the most relevant 15000 features using the chi-square test. Our feature vectors were then constructed using the TF-IDF weighting scheme.[2]

4.3 Evaluation Strategy and Evaluation Criteria

We define the duration of a timepoint to be a week and maintain a sliding window of five weeks. We perform prequential evaluation [13]: we first test on all documents for the incoming week and then adapt by using only the sampled documents whose labels have been provided by the oracle.

As we use entropy to calculate our confidence measure, we evaluate on log loss decrease [2]. The log loss l_t at timepoint t is given by,

$$l_t = -\frac{1}{|U_t|} \sum_{i \in U_t} \sum_{j \in C} b_{ij} \log p_{ij} \tag{3}$$

where b_{ij} is a binary indicator of whether or not label j is the correct classification for instance i, and p_{ij} is the model probability of assigning label j to instance i.

5 Experimental Evaluation

As the Stochastic Gradient Descent classifier was found to be effective for sentiment analysis in [15], we used the same with hinge loss, *l2* penalty and alpha value of 0.0001 to optimize the objective function of a linear support vector machine as our *base learner*. The base learner was calibrated using Platt Scaling to obtain probabilistic outputs.

[2] Code and supplementary material available at https://github.com/elrasp/osm

For each timepoint t, we set a fixed budget $\mathcal{B} = 0.1$. For the query strategies, we set the fixed threshold $\theta = 0.9$, and the suggested values of 0.01 and 1 for the threshold adjustment step s and the variance of the normally distributed random number generator δ respectively [11].

In Section 5.1, we describe the underlying class distribution of the data stream for these datasets and in Section 5.2, we analyze the influence of the oracle's availability on the performance of the learner.

5.1 Distribution of Data

The weekly underlying class distribution of data in the stream of opinionated documents for the Yelp and Amazon datasets are shown in Figure 3(a) and Figure 3(b) respectively.

Yelp: For the Yelp dataset, we observe a gradual increase in the number of reviews obtained over time. The proportion of neutral reviews received remain almost constant for the entire data stream. In comparison, the positive and negative reviews are always increasing. Also, we find that the positive reviews dominate the class distribution accounting for more than 50% of the reviews in any week.

Amazon: Unlike the Yelp dataset, the amazon dataset exhibits sudden bursts in the volume of reviews received. This occurs as some chosen product categories are more popular than others and receive more reviews. We also observe that mostly there is a burst in the positive reviews received as compared to the negative and neutral reviews. Similar to the Yelp dataset, the positive reviews dominate the class distribution.

5.2 Impact of the Oracle's Availability on Learning

Figure 2 shows how the oracle availability, simulated by the method of Section 4.1, affects the number of queries answered. We vary the availability between 1.0 (all queries answered) and 0.1 (only 10% of the queries per batch are answered) in steps of 0.1.

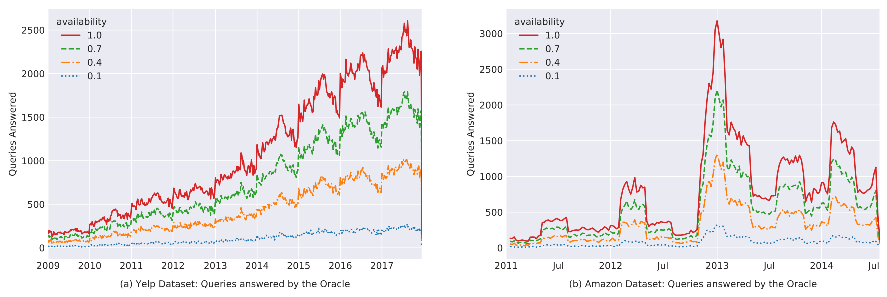


Fig. 2. Queries answered by the Oracle for varying availabilities

In Figure 3 we show the results of evaluation for the Yelp (on the left) and Amazon (on the right) datasets. We aggregated the log loss results for all the timepoints every two months and plot the mean (lines) and standard deviation (colored area around the line). For the Yelp dataset, in general, we observe that the error in performance gradually reduces as the stream progresses irrespective of the oracles availability and the query strategy used. On the other hand, the learner finds it much more difficult to adapt to the evolving data stream of the Amazon dataset.

In the early stages of the stream, where the data volume in the stream is low, we observe the learner performing better for lower oracle availabilities. As more and more data is accumulated, the need to have the oracle for the Yelp dataset drops but remains for the Amazon dataset as it exhibits more drift.

We further conducted experiments with oracle availabilities varying between 0.01 and 0.1 in steps of 0.01 and compared the different query strategies at varying availabilities with the non-parametric Friedman's test followed by Nemenyi post-hoc [16]. Friedman's test proceeds by ranking the models under consideration. The best performing model is given the rank 1, the second best 2 and so on. If two or more models have identical performance they are given an average rank. The null hypothesis of Friedman's test states that all the models perform the same and thus, will have the same average rank. If the test rejects the null hypothesis, we proceed with the Nemenyi post-hoc test that makes pair-wise comparisons of the different models. It identifies statistically significant models if the difference between their average rank is more than the critical distance.

Figure 4(a) and Figure 4(b) shows the critical distance diagram of the Yelp and Amazon dataset respectively. In these diagrams the better performing models are ranked higher and are placed to the right. The model name corresponds to a combination of the query strategy and the value of the oracle availability, whereupon "rand" in the name refers to the random strategy. The models whose difference in average rank is less than the calculated critical distance (CD) are connected to each other by a horizontal line, indicating that these models are statistically indifferent to each other.

As we can see in these figures, there are strategies that which perform significantly better when the oracle availability changes. At very low oracle availabilities we find that there is no strategy that performs significantly better than the others. For the Amazon dataset, even at higher oracle availabilities there is no difference in the performance of the strategies. This could mainly be attributed to the nature of the drift exhibited in both the datasets.

6 Conclusions

The need for an oracle depends on the overall variability of the dataset. If there is convergence over time as in the case of the Yelp dataset, the need for an oracle is limited, because the learner can predict the labels by itself. Hence, low availability of the oracle is only relevant if there is drift.

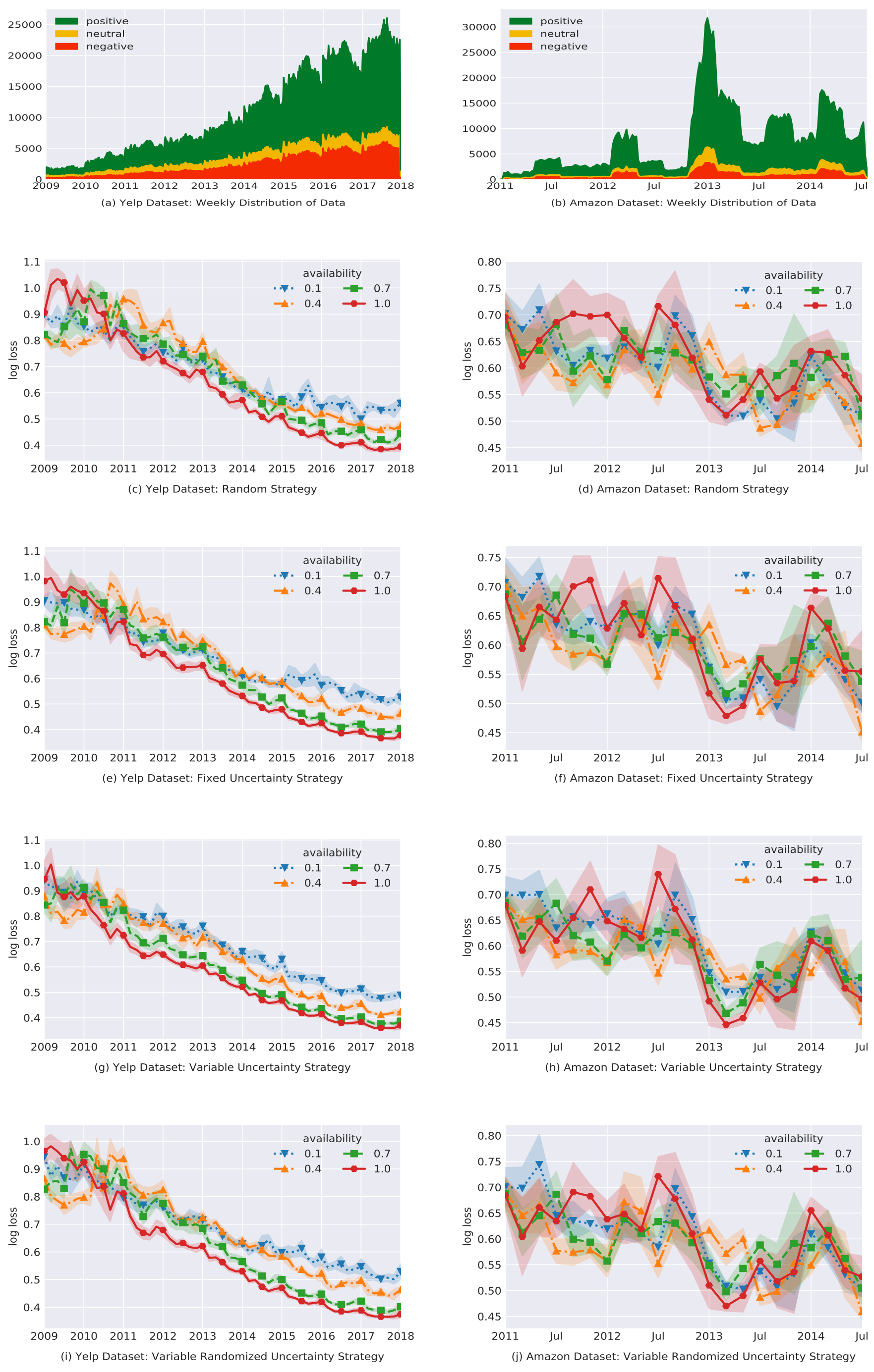
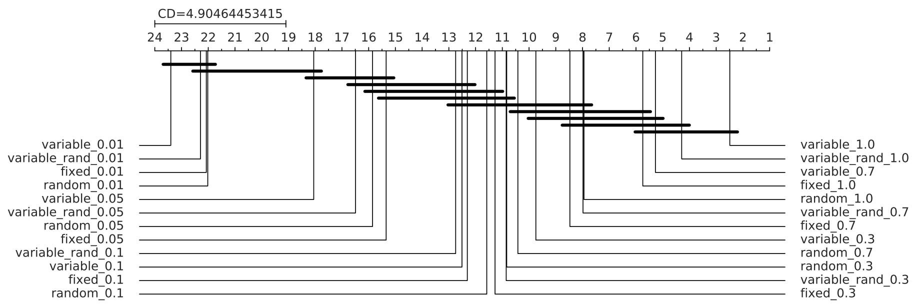
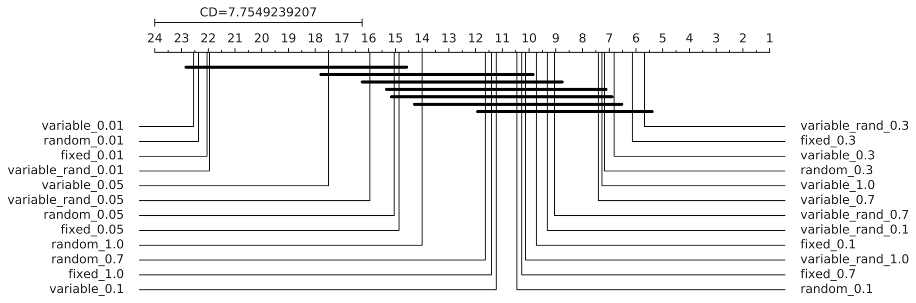


Fig. 3. Evaluation results of the Yelp (on the left) and Amazon (on the right) datasets



(a) Yelp Dataset



(b) Amazon Dataset

Fig. 4. Critical Distance Diagram. The models can be identified by the query strategy and oracle availability. The best performing models are shown to the right and models that are not statistically different to each other are connected by the horizontal line.

If an oracle is available, the random strategy is a good choice, as it shows the same tendency as the other strategies, is easy to implement and is fast. However, more experiments are needed to check whether stronger active learning strategies can beat the random sampler in this setting, by capitalizing more effectively on the few available labels. Experiments are also required for different domains.

To improve model quality, we intend to consider more elaborate querying strategies [17], and to investigate whether instance-based active learning might deliver better results than the block-based active learning paradigm we currently use.

To compensate for oracle inavailability, we also intend to combine active learning with semi-supervised learning. Semi-supervised methods are used to propagate labels to the arriving instances, cf. [18], [19]. In that case, we want to investigate how disagreement between oracle and self-learner can be alleviated in a seamless way.

Acknowledgements. This work was inspired and partially conducted (last author) within the project OSCAR Opinion Stream Classification with Ensembles and Active Learners, funded by the German Research Foundation.

References

1. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)
2. Fu, Y., Zhu, X., Li, B.: A survey on instance selection for active learning. Knowledge and Information Systems **35**(2) (May 2013) 249–283
3. Lughofer, E.: On-line active learning: A new paradigm to improve practical useability of data stream modeling methods. Information Sciences **415-416** (nov 2017) 356–376
4. Wu, W., Liu, Y., Guo, M., Wang, C., Liu, X.: A probabilistic model of active learning with multiple noisy oracles. Neurocomputing **118** (2013) 253 – 262
5. Donmez, P., Carbonell, J.G.: Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management. CIKM '08, New York, NY, USA, ACM (2008) 619–628
6. Shickel, B., Rashidi, P.: ART: an availability-aware active learning framework for data streams. In Markov, Z., Russell, I., eds.: Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016, Key Largo, Florida, May 16-18, 2016., AAAI Press (2016) 92–97
7. Zimmermann, M., Ntoutsi, E., Spiliopoulou, M.: Adaptive semi supervised opinion classifier with forgetting mechanism. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing. SAC '14, New York, NY, USA, ACM (2014) 805–812
8. Zimmermann, M., Ntoutsi, E., Spiliopoulou, M.: Incremental active opinion learning over a stream of opinionated documents. WISDOM'15 (Workshop on Issues of Sentiment Discovery and Opinion Mining) 2015 at Knowledge Discovery and Data Mining, KDD'15 Workshops 2015, Sydney, Australia, August 10, 2015 (2015)
9. Žliobaitė, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with evolving streaming data. In Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M., eds.: Machine Learning and Knowledge Discovery in Databases, Berlin, Heidelberg, Springer Berlin Heidelberg (2011) 597–612
10. Žliobaitė, I., Bifet, A., Holmes, G., Pfahringer, B.: Moa concept drift active learning strategies for streaming data. In Diethe, T., Balcazar, J., Shawe-Taylor, J., Tirnauca, C., eds.: Proceedings of the Second Workshop on Applications of Pattern Analysis. Volume 17 of Proceedings of Machine Learning Research., CIEM, Castro Urdiales, Spain, PMLR (19–21 Oct 2011) 48–55
11. Žliobaitė, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drifting streaming data. IEEE Transactions on Neural Networks and Learning Systems **25**(1) (Jan 2014) 27–39
12. Smailović, J., Grčar, M., Lavrač, N., Žnidaršič, M.: Stream-based active learning for sentiment analysis in the financial domain. Inf. Sci. **285**(C) (November 2014) 181–203
13. Gama, J.a., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4) (March 2014) 44:1–44:37
14. McAuley, J., Targett, C., Shi, Q., van den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 2015, ACM Press (2015)
15. Bifet, A., Frank, E.: Sentiment knowledge discovery in twitter streaming data. In: Proceedings of the 13th International Conference on Discovery Science. DS'10, Berlin, Heidelberg, Springer-Verlag (2010) 1–15

16. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7** (December 2006) 1–30
17. Kottke, D., Krempl, G., Spiliopoulou, M.: Probabilistic active learning in datastreams. In Fromont, E., De Bie, T., van Leeuwen, M., eds.: Advances in Intelligent Data Analysis XIV, Cham, Springer International Publishing (2015) 145–157
18. Dyer, K.B., Capo, R., Polikar, R.: COMPOSE: A semisupervised learning framework for initially labeled nonstationary streaming data. IEEE Transactions on Neural Networks and Learning Systems **25**(1) (jan 2014) 12–26 Journal Article Research Support, U.S. Gov't, Non-P.H.S.
19. Souza, V.M.A., Silva, D.F., Gama, J., Batista, G.E.A.P.A.: Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In Venkatasubramanian, S., Ye, J., eds.: Proceedings of the 2015 SIAM International Conference on Data Mining. [Society for Industrial and Applied Mathematics] (jun 2015) 873–881