Batch-based Active Learning: Application to Social Media Data for Crisis Management

Daniela Pohl[a,*], Abdelhamid Bouchachia[b], Hermann Hellwagner[a]

[a] *Institute of Information Technology, Alpen-Adria-Universität Klagenfurt, Universitätsstr. 65-67, Klagenfurt, Austria*
[b] *Smart Technology Research Center, Bournemouth University, Poole House Talbot Campus, Fern Barrow Poole, BH12 5BB, Bournemouth, UK*

Abstract

Classification of evolving data streams is a challenging task, which is suitably tackled with online learning approaches. Data is processed instantly requiring the learning machinery to (self-)adapt by adjusting its model. However for high velocity streams, it is usually difficult to obtain labeled samples to train the classification model. Hence, we propose a novel **o**nline **b**atch-based **a**ctive **l**earning algorithm (OBAL) to perform the labeling. OBAL is developed for crisis management applications where data streams are generated by the social media community. OBAL is applied to discriminate relevant from irrelevant social media items. An emergency management user will be interactively queried to label chosen items. OBAL exploits the boundary items for which it is highly uncertain about their class and makes use of two classifiers: k-Nearest Neighbors (kNN) and Support Vector Machine (SVM). OBAL is equipped with a labeling budget and a set of uncertainty strategies to identify the items for labeling. An extensive analysis is carried out to show OBAL's performance, the sensitivity of its parameters, and the contribution of the individual uncertainty strategies. Two types of datasets are used: synthetic and social media datasets related to crises. The empirical results illustrate that OBAL has a very good

*Corresponding author; phone: +43 463 2700 3688; fax: +43 463 2700 993688
Email addresses: daniela@itec.aau.at (Daniela Pohl), abouchachia@bournemouth.ac.uk (Abdelhamid Bouchachia), hellwagn@itec.aau.at (Hermann Hellwagner)

discrimination power.

Keywords: Online Learning, Active Learning, Classification, Social Media, Crisis Management

1. Introduction

In the presence of less labeled data for training a classifier, active learning can be applied to interactively query the user about the label of an input. Active learning has been the subject of intensive investigation over the last decade Zhu et al. (2007). However, most of the work has focussed on offline active learning Nguyen & Smeulders (2004); Cohn et al. (1996). Indeed, few attempts have been made to develop online active learning algorithms for data streams Ienco et al. (2013); Attenberg & Provost (2011).

By its very nature, active learning and in particular stream-based active learning is relevant to various applications where external feedback from the environment is used to enhance the classification performance. Learning from social media (SM) data for a particular application like crisis management may require active involvement of the users who could be emergency management staff members (e.g. first responders) to label ambiguous items. Interestingly, in recent years, SM has become a well-established communication medium for the public to exchange information. Mobile devices and mobile Internet allow users to publish information almost anywhere at anytime. This makes SM a very important source of information for different purposes. There is considerable effort by the research community to harness social media for emergency management. Many studies in the context of SM and emergency management show the usefulness of this data for emergency preparation, response planning, and recovery strategies Pohl et al. (2013); Pohl (2014). More precisely, emergency departments have already noticed the importance of social media for gathering, monitoring, and disseminating information Denef et al. (2013); Hughes et al. (2014) but they mostly use simple built-in browsing and search mechanisms as for Twitter.

To exploit SM in the context of crisis management, it is necessary to identify data that is relevant to the crisis/emergency at hand. Hence, appropriate preprocessing is necessary in order to distinguish between relevant and irrelevant items by involving the emergency management staff. Such professionals can then share their experience and knowledge to develop useful learning systems.

In this paper, we propose a ***b**atch-based **a**ctive **l**earning algorithm (OBAL).* This algorithm uses *label uncertainty* Cohn et al. (1994) as query criterion. It does that by examining the boundary separating the classes, which represent uncertainty regions. Here, the classes are "relevant" and "not relevant". OBAL self-adapts in response to changes in the data stream (i.e., shifts in the boundary region) by continuously updating the boundary information.

Different uncertainty strategies acting on the boundary information are proposed and applied to decide which data items need to be queried. Moreover, while OBAL can use any classifier, in this paper we use the k Nearest Neighbors (kNN) and Support Vector Machine (SVM). We also introduce a number of query strategies to request feedback from the user. But to limit the number of queries, a labeling budget is used.

The paper is structured as follows. Section 2 gives an overview of the related work. Section 3 presents the details of OBAL and defines the concept of boundary items and how they are determined. Section 4 describes the different query strategies used to identify ambiguous items. Section 5 presents the concept of budget. Section 6 describes the experiments and discusses the results. Finally, Section 7 concludes the paper.

2. Related Work

Research related to this work is concerned with (1) online learning from social media especially in crisis management, (2) active learning, and (3) active learning with a budget. The areas are covered in the following sections.

2.1. Learning from Social Media in Crisis Management

Social media for crisis management is becoming increasingly popular. Several approaches deal with online learning based on SM in crisis management. For example, Yin et al. Yin et al. (2012) identify topics online from Twitter during a crisis using textual similarity. Klein et al. Klein et al. (2012) analyze tweets with the help of textual and network-based analysis. Starbird Starbird (2011) introduces a predefined grammar for tweets to support the automatic analysis of new incoming tweets. Sheth et al. Sheth et al. (2011) identify events by using event descriptors.

Twitcident, introduced by Abel et al. Abel et al. (2012), uses manual inserted keyword rules to analyze Twitter streams in time. MacEachren et al. MacEachren et al. (2011) utilize named entity recognition for event identification in tweets. Other tools use several visual analytic approaches to identify important topics during emergency. For example, TweetTracker from Kumar et al. Kumar et al. (2013) apply named entity recognition to identify important topics. Also, TweetXplorer introduced by Morstatter et al. Morstatter et al. (2013) makes use of visual analytics to identify tweets during emergency situations.

Li et al. Li et al. (2012) employ commonly used keywords from similar emergency situations to train a classifier to identify crises. Zhou et al. Zhou et al. (2013) classify incoming microblogs to redirect them to the emergency agency in charge (e.g., police, fire department). Imran et al. Imran et al. (2014b) automatically classify incoming tweets into information categories, for example, damage, injured, etc.

Besides Twitter, other platforms are also considered in online analysis. For example, Maxwell et al. Maxwell et al. (2012) introduce a system called Crisees for sentiment analysis based on Twitter and YouTube streams. Petkos et al. Petkos et al. (2014) determine events online based on Flickr streams by considering community detection algorithms. Papadopoulos et al. Papadopoulos et al. (2013) use geo-referenced and visual information from SM data (e.g., Twitter, Flickr, YouTube) to aggregate related items for situational awareness.

2.2. Active Learning

Active learning for data streams is used for collecting labels of unlabeled data examples to enhance the accuracy of the classifier. There are two classes for active learning: *pool-based* and *streaming-based* methods. Most of the methods are pool-based; only few methods are dedicated to data streams.

Žliobaitė et al. Žliobaitė et al. (2014) apply different uncertainty strategies to query items. Smailović et al. Smailović et al. (2014) use active learning for sentiment analysis using Twitter. The idea is to predict stock market movements. Zhu et al. Zhu et al. (2007) describe an active learning algorithm for data streams based on classifier ensembles. The idea is to label those items that cause a high variance in the classification results of the ensemble. Ienco et al. Ienco et al. (2013) use a pre-clustering step in order to identify relevant items to be labeled by the user. Therefore, the data stream is partitioned into segments and each segment is clustered. The homogeneity of each resulting cluster is examined based on the classifier results for each item within the cluster. The items of those clusters that contain different labels are chosen, since those are the most uncertain ones.

AIDR Imran et al. (2014b) is used to automatically classify incoming tweets during a crisis into different classes, for example, in caution/advise, damage, casualties, etc. The classification model used in the background utilizes random forests (i.e., decision trees) with pure textual features (i.e., unigrams and bigrams extracted from the text) Imran et al. (2014a). An active learning approach is used, where members of a crowd label tweets. It is described that with active learning, items that are selected *"are close to the decision boundary and for which the labels are maximally informative"* Imran et al. (2014a). A small subset of SM items is selected for requesting labels Imran et al. (2014b). The PyBossa crowdsourcing platform is used to contact volunteers for labeling the data.

There are also other approaches related to SM and active learning. Chatzilari et al. Chatzilari et al. (2014) use active learning to categorize pictures fetched from Flickr. In this context, the labeling of an image is performed through

user-defined tags assigned to images instead of a human deciding explicitly on the labels. They use an active learning approach based on SVM for category classification. Zhuang et al. Zhuang et al. (2011) use active learning in assessing social strength between users. They select training examples using active learning to build pairs considering different similarity measures (e.g., geo-location, similar friendship, interest groups, etc.) Zhuang et al. (2011). Jin et al. Jin et al. (2011) introduce a spammer detection algorithm for social networks based on active learning. Several features, such as image content, text features, and profile information are extracted from posts which are the basis for classifying a profile as spammer or normal user.

Based on the active learning categorization by Settles et al. Settles (2010), our approach is a "stream-based selective sampling approach" based on updated boundary items and considering different strategies to request instances for labeling.

2.3. Concept Drift

A crisis is a dynamic environment where situation changes over time. Within such a dynamic environment, concept drifts are inherent. In general, a concept drift describes changes within incoming data, which forces a learning algorithm to adapt to the changing data, see Webb et al. (2016).

Li et al. Li et al. (2016) consider concept drift to predict the popularity of social media items within a social media network. For the prediction, a classification ensemble is used and offline trained based on different time intervals. Gama et al. Gama et al. (2014) summarize concept drift approaches and state sentiment analysis for social media monitoring as valuable application. Another approach for coping with concept drifts in textual data streams is suggested by Song et al. (2016). They use Clustering Trees, i.e. a Clustering Forest, as base to classify items in the stream.

Mohamad et al. Mohamad et al. (2017) suggest an active learning algorithm considering concept drifts based on Growing Gaussian Mixture Models and online logistic regression. The approach uses two selection criteria to iden-

tify instances for labeling, namely density based and uncertainty based criteria. Borchani et al. Borchani et al. (2015) introduce a stream-based Bayesian classifier for the financial sector. They conducted, as it is also performed in the paper at hand, experiments based on real-world and synthetic datasets. Guo and Liew Guo & Liew (2016) use concept drift tracking for time series analysis. After a concept drift is detected, the online training is performed to adapt to the new situation. Da Costa et al. da Costa et al. (2016) propose concept detection based on unsupervised machine learning methods, e.g., hierarchical clustering. Another approach, proposed by Pozzolo et al. (2015), uses concept drift for fraud detection of credit cards. The approach uses Balanced Random Forest as classifier.

In our approach, concept drift within the data is considered due to a continuous update of the boundary vectors. Currently, the classification is based on kNN and SVM. It can be changed to any other model acting on the boundary data.

2.4. Active Learning with Budget

A budget can be used to limit the number of queries the user has to handle. The budget can be derived from time constraints, budget money, etc. This prevents the user from doing too much labeling. For example, Žliobaitė et al. Žliobaitė et al. (2014) consider a budget over data streams based on a moving average. They utilize several uncertainty strategies to ask the user for feedback. They use a random budget. Ienco et al. Ienco et al. (2014) apply high-density region and posterior probability calculation to identify items to be queried.

Dasgupta and Hsu Dasgupta & Hsu (2008) introduce also a budget into their active learning approach. The budget controls the number of queries per batch. The authors apply a hierarchical clustering approach to select items for labeling per batch. Imran et al. Imran et al. (2014a) use a fixed number of labels to request as budget limit. In addition, they test different budget usage strategies (e.g., all labels at the beginning or separated between periods). Attenberg and Provost Attenberg & Provost (2011) describe a budget approach for active

learning which is based on the usefulness of incoming items considering online density estimation (i.e., the possible repetition of an item within the stream). Also, Cesa-Bianchi et al. Cesa-Bianchi et al. (2006) use a label request strategy for online learning based on results of the perceptron used for classification (i.e., depending on the margin). The upper bound of the number of labels is based on the margin itself.

The approach presented in Vijayanarasimhan et al. (2010) shows the application of a budget on image and video recognition. In this case, the budget is defined based on the funding available (e.g., amount of money spent on the *Mechanical Turk*[1] platform). The items to label consume different amounts of the budget depending on the complexity involved in labelling them. Their selection is viewed as an optimization problem.

For our experiments, we adopt the approach given in Žliobaitė et al. (2014) in order to limit the number of user queries. We also implement different uncertainty strategies to decide on the ambiguity of the classification results, as described later in Section 4.

3. Batch-based Active Learning

OBAL makes use of the uncertainty criterion which stipulates that the data instances which the model is least certain about their labels are queried. Usually, the most uncertain items typically lie close to the classification boundary (see Fig. 2). Thus, training the classifier on those items is expected to adjust the boundary, achieving therefore better classification accuracy. Figure 1 shows the general processing steps of OBAL. While OBAL can be used for any type of input, in our case, it is applied to textual data (social media items). The aim is to distinguish between relevant items and and irrelevant ones (i.e., $label \in \{relevant, irrelevant\}$). The textual items are first transformed into a vector space model, where each feature is captured by the *term frequency-inverse*

[1] http://www.mturk.com/

document frequency (tf-idf) Manning et al. (2008); Pohl et al. (2012).

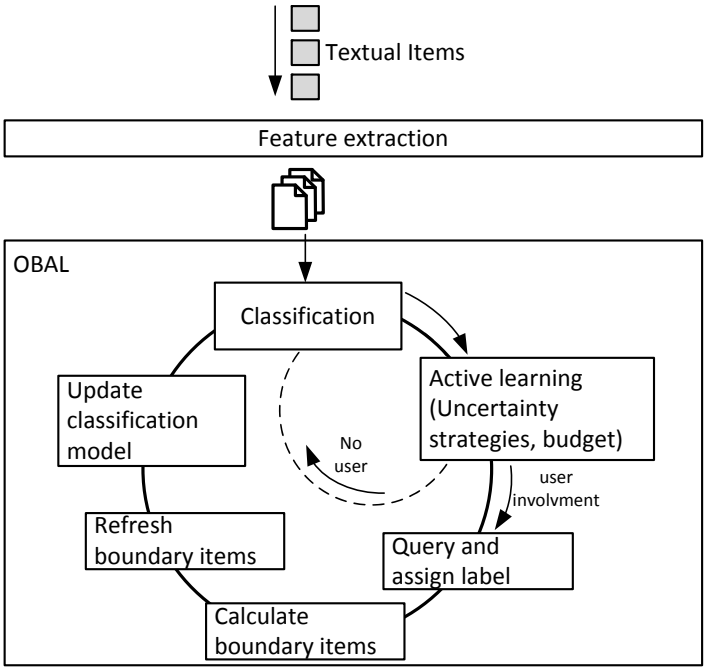


Figure 1: Workflow of OBAL

OBAL consists of the following protocol:

1. For each new data item
2. If the classifier (trained on the labeled data) is *uncertain* about the label of the item according to some uncertainty criteria - see Section 4, then
   (a) Query the label of the new input provided that enough budget is available (see Section 5).
   (b) Check boundary condition for the item and, if satisfied, add the item to the boundary pool of labeled samples.
   (c) Train the classifier using the new pool.

In this paper, two classifiers (*kNN* and *SVM*) are used. Any other classifier can be chosen though. Each classifier has its own way of using the boundary items in classification. For kNN, an item is considered as uncertain if there is no

label majority in its neighborhood of boundary items (see uncertainty strategies in Section 4.1). On the other hand, SVM is trained on the known boundary items (see Fig. 2) and will be continuously updated as new boundary items are identified. If SVM is not certain about the label of that sample (see uncertainty strategy in Section 4.2), the label of that sample is queried. The identified boundary items are used to (re-)train the classifier.

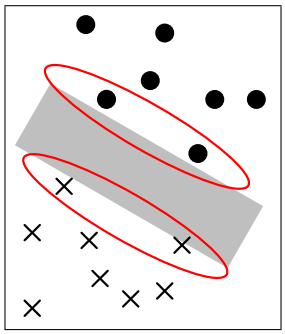


Figure 2: Boundary items from two classes: '•' and 'x'

Table 1 introduces the list of symbols used in the rest of this paper. Algorithm 1 shows the steps of OBAL. The algorithm is divided into two phases: *initial phase* and *execution phase*. The *initial phase* is used for a "cold-start" to build an initial classification model (see Algorithm 1 step 2). In our experiments, we have used a minimum number of $\zeta = 3 * C$ (3 examples from each class), where C is the number of classes. If there is enough data (see step 5 of Algorithm 1), the boundary examples are computed and if there are enough boundary examples, the second phase, the execution phase, can start.

Algorithm 2 proposed in Bouchachia (2014) shows how the boundary samples are computed using Gabriel graph. Two samples with different labels are part of the boundary, if there is no other sample within the diameter of these two samples (see steps 6-9). Figure 3 illustrates graphically how the boundary samples are computed in Algorithm 2. Note that the algorithm uses the *Euclidean* distance expressed as: $dist(\mathbf{v}, \mathbf{x}) = \left(\sum_{i=1}^{M}(v_i - x_i)^2\right)^{\frac{1}{2}}$

During the *execution phase*, the boundary vectors are used to build a classi-

Table 1: List of symbols

Symbol for OBAL	**Description**
C	Number of classes.
X	$X = \bigcup X_i$ for $i = 1, \ldots, C$ is the set of labeled items, where X_i describes the labeled items belonging to class i.
ω	Maximum number of labeled examples in X, e.g., $\lvert X \rvert = \omega = 50$.
$Boundary$	Describes the current boundary vectors; $Boundary$ is a $C \times C$ array, where $Boundary(i, j)$ is the set of boundary examples between class i and class j.
ζ	Minimum number of boundary examples per class (threshold) to be considered in the initial phase (cold start). $\zeta \leq \omega$.
uRes	$uRes.uncertainty = true$ if the classification of the input is uncertain otherwise $uRes.uncertainty = false$. $uRes.label$ contains the predicted label of the current item.
k	Number of nearest neighbors in kNN.
svm_start	Boolean to indicate whether SVM can be trained.
svm_update	Boolean whether SVM has been updated.
$svm\_threshold$	Maximum distance of data to the hyperplane in SVM.

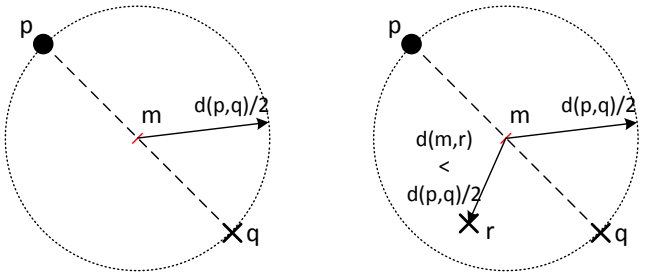


Figure 3: Boundary samples for two classes ('•' and 'x'); *left:* p and q as boundary samples, *right:* p and q are not boundary samples because there is another sample between them.

fier in order to predict the label of the new incoming inputs (see Algorithm 1, step 12).

3.1. Classification

Although any classifier can be used in Algorithm 1 step 12. We apply two classifiers in this paper: kNN and SVM. The kNN operates on the current known boundary vectors (see Algorithm 3 and Table 1). The kNN results in k neighbors for the current input, where each neighbor consists of the vector and the assigned label representing it. Depending on the results of the kNN, several uncertainty strategies (see Section 4.1) are implemented to identify uncertainty in classification outputs (see Algorithm 3, step 2).

For the SVM, the different boundary vectors are used to update the classifier (Algorithm 4 and Table 1). The SVM is initialized after the *initPhase* (i.e., $svm\_start = 1$) or updated after *svm_update* user queries (e.g., every 6 queries). The uncertainty strategy of SVM focuses on the distance between the input and the margin of the SVM border (see Section 4.2). We used SVM with the "Gaussian Radial Basis Function" kernel.

If there is an uncertainty about the classification result (kNN or SVM) and enough budget available (see Section 5), the user is asked to provide a label. Only labeled data is used to update the boundary vectors during execution. This allows us to respect drifts within the data. After the feedback of the user,

the boundaries are calculated and the necessity for an update/cleaning step is checked.

After ω items were processed, the update step is performed (see Algorithm 5). First, all non-boundary vectors are removed from X. If the number of remaining vectors is higher then the threshold ω, all remaining vectors - except the most recent boundary vectors - are removed from X so that ω vectors remain.

4. Query Strategies

The following sections describe the uncertainty strategies that allow deciding when to query data items.

4.1. kNN Query Strategies

We implemented four types of uncertainty estimation based on kNN results. The uncertainty estimation is based on a voting concept depending on the labels and/or distances of the neighbors of the current input:

- *Majority vote (MV):* It considers the majority of class labels in the kNN result (see Algorithm 6). If there is no majority in favour of one class, the user is asked to label the input.

- *Distance-based maximum vote (DMV):* This strategy is similar to *MV*, but relies to the distance to neighbors (see Algorithm 7). If the distance of the input to most of the neighbors is above the average distance, it is supposed that the current input is too far away to make an acceptable decision.

- *Two-thirds majority vote (TMV):* This means that more than two-thirds of the neighbors must have the same label (see Algorithm 8). If at least 2/3 of the neighbors do not have the majority, the user is asked to provide a label.

- *Weighted vote (WV):* This strategy was suggested by Dudani Dudani (1976) (see Algorithm 9). In this case, neighbors closer to the *input* of one class are higher weighted than other ones. If the average difference between the weighted distance of all pairs of classes is lower than a given thresholds ε, the user is asked to provide a label. The assumption is that the calculated weighted distances are too close to make a decision.

4.2. SVM Query Strategy

For SVM, we make use of the margin between the boundary vectors of two classes recognized in the SVM. If the distance to the hyperplane is below a predefined threshold ($svm\_threshold$), the classification result is considered as uncertain (see Algorithm 10 for details).

5. Budget

The idea of active learning is to query uncertain data. To limit the number of user queries, a *budget* may be defined. The budget is the maximum number of labeling queries for feedback. In this work, we adapt the budget method presented in Žliobaitė et al. (2014). The budget b_t consumed after t queries is defined as follows Žliobaitė et al. (2014):

$$\begin{aligned} b_t = \frac{u_t}{s}, where: \\ u_t = \lambda u_{t-1} + labeling_t \\ \lambda = (s-1)/s \end{aligned} \tag{2}$$

where u_t is the number of queries made in the last s steps. The window s acts as memory (e.g., last 100 items Žliobaitė et al. (2014)). λ describes the fraction of including value u_{t-1}. $labeling_t$ updates u_t based on the requested label (i.e., $labeling_t = 0$ if no label was queried and $labeling_t = 1$ if there was a label requested) at the current item t.

One needs to improve a threshold B on b_t, that is the upper bound of the number of queries and the fraction of data from window bw that can be labeled

(i.e., $B = 0.2$ corresponds to 20%). At each step, one input is processed. The *isBudgetAvailable()* procedure in Algorithm 1 checks, if enough budget is available (i.e., $b_t < B$). If so, the uncertain input is queried.

6. Experiments

In this section, we present the datasets, show the experimental settings, and discuss the evaluation results.

6.1. Datasets

The experiments have been performed on two kinds of datasets: *synthetic datasets* to understand the general performance of OBAL in controlled settings, and *real-world social media datasets* to investigate the behavior on textual data. In the following, a short description of both classes of datasets as well as the performance of the proposed classifier is introduced.

6.1.1. Synthetic Datasets

Two datasets are used: (1) The *synthetic dataset (SD)* consists of 4 batches with 200 data items. Figure 4 illustrates the synthetic dataset. Each batch is divided into two classes, with 100 items belonging to the relevant class (denoted as 'o') and 100 to the irrelevant class (denoted as 'x'). The data is randomized sampled and sequentially presented to OBAL. It allows us to study the behavior and performance of the algorithm.

(2) The *non-contiguous class synthetic dataset* (NCSD) is presented to OBAL as one batch (see Figure 5), i.e., the data is selected randomized and sequentially presented to OBAL. The NCSD consists of three clusters: two clusters of class 'o', separated by class 'x'. Each partition consists of 200 data items. It allows us to study the behavior and performance of the algorithm when a non-contiguous class is given.

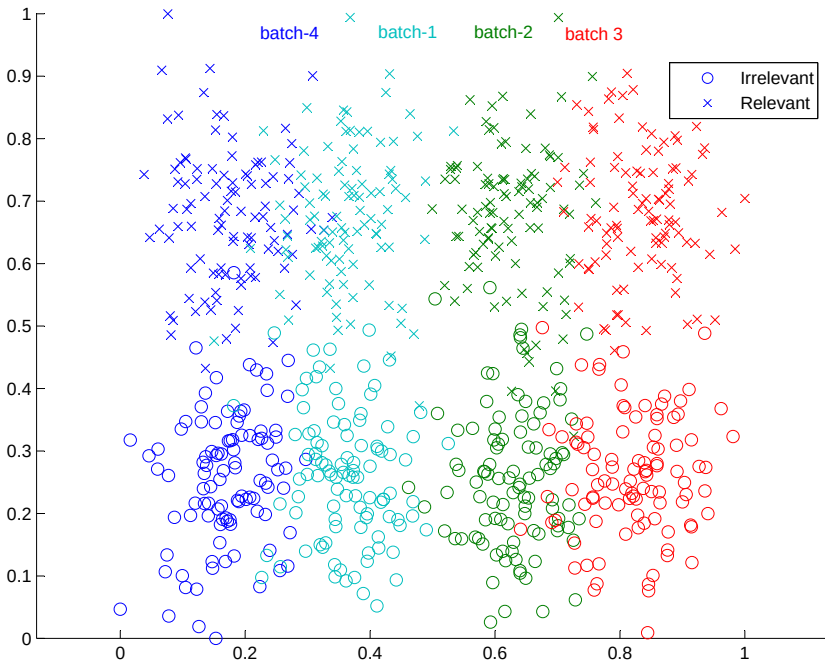


Figure 4: Synthetic dataset (SD) presented to OBAL as batch-1, batch-2, batch-3, and batch-4.

6.1.2. Real-World Datasets

We used two real-world datasets from the CrisisLexT26 collection Olteanu et al. (2015). The collection contains Twitter data from different crises around the world. In detail, we used the Colorado Floods (CF) and Australian Bushfires (AB) datasets. For one crisis, 1000 items are randomly selected and labeled via a crowdsourcing platform to identify the relevance of the items. Items are marked in four categories: *related to the crisis and informative*, *related to the crisis - but not informative*, *not related*, and *not applicable*. We consider items only as "relevant" when they are labeled as *related to the crisis and informative*, otherwise they are irrelevant.

For CF, we examine the period between "2013-09-12 07:00:00" and "2013-09-29 10:00:00" since this period contains most of the items/tweets. CF contains 751 relevant and 224 irrelevant items and in sum approx. 189 repetitions of

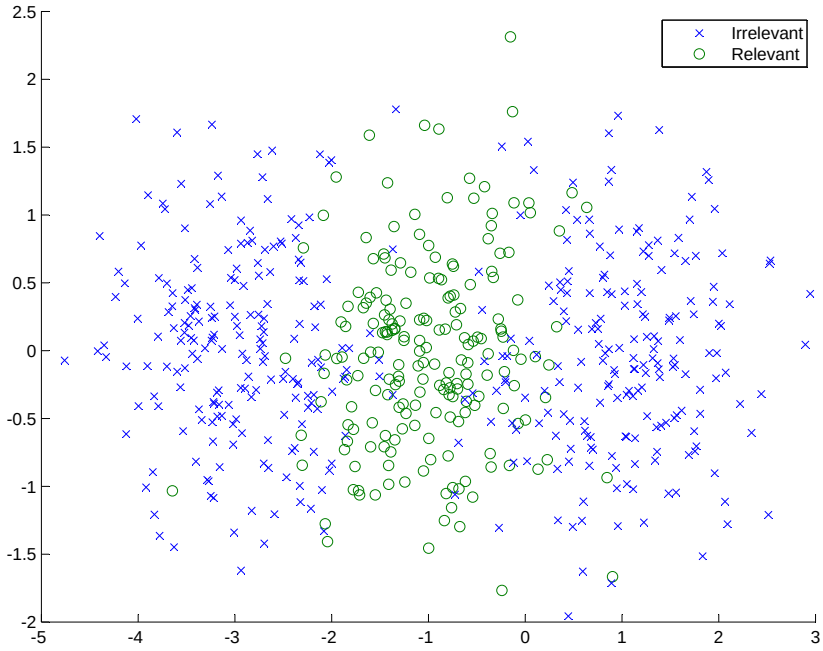


Figure 5: Non-contiguous synthetic dataset (NCSD) presented to the OBAL as one batch.

text messages[2]. Based on the Levenshtein distance ($ldis$) Levenshtein (1966), there exist 105 items with similar text (i.e., $ldis \leq 0.2$), which is a quite small number. This also indicates that the lengths of the repeating text fragments are very small (105 vs. 189 repetitions of text).

For the AB dataset, we cover the relatively long period between "2013-10-17 05:00:00" and "2013-10-29 12:30:00" due to its low item density. It has the following characteristics: 645 relevant, 408 irrelevant items with approx. 385 repetitions of text messages[2]. The AB dataset has a high amount of similar items, i.e., based on $ldis \leq 0.2$, there exist 582 items with similar text. Due to the increased similarity of the items, it is more difficult to distinguish between relevant and irrelevant items. There are some overlapping items between relevant and irrelevant items, increasing the effort to distinguish. Hence, AB

[2]Repetitions are retweets. It is an approximation since there is no mandatory retweet syntax Boyd et al. (2010).

is an interesting dataset in order to test the approach under hard conditions. For instance, the following example shows two differently labeled items with the same content Olteanu et al. (2015):

- Wed Oct 16 17:12:46 +0000 2013: "390525739864702976",***"RT @Xxxx: A dog has risked its life to save a litter of newborn kittens from a house fire in Melbourne, Australia http://t.co/Gz.."***,Eyewitness,Affected individuals,**Related and informative**

- Wed Oct 16 17:13:57 +0000 2013: "390526037689630720",***"RT @Xxxx: A dog has risked its life to save a litter of newborn kittens from a house fire in Melbourne, Australia http://t.co/Gz..."***,Not labeled,Not labeled,**Not related**

We created 30-minutes batches for both datasets which is a reasonable time period for planning in crisis management Pohl et al. (2015). There are also inactive periods within the datasets.

6.2. Evaluation Metrics

For evaluation, we used two metrics. First, the *error-rate (ER)* (see Eq. 3) is calculated based on the label assignments of the N items in the dataset L compared to the ground truth G. $|L \neq G|$ is the number of items, where the assigned labels L do not agree with the ground truth G. The average ER based on all batches is calculated and used to estimate the performance.

$$ER(L;G) = \frac{|L \neq G|}{N} * 100 \tag{3}$$

Second, for assessing the influence of the budget setting on the algorithm, we also consider the *average number of user-queries (Q)* per batch. Both, ER and Q are visualized within diagrams based on the experimental setting given in the next section.

6.3. Results

We conducted an extensive analysis of OBAL's performance and the sensitivity of the used parameters. This section covers the results of the analysis and discusses the results for the 2-dim synthetic datasets (SD and NCSD) and the

real-world datasets (CF and AB). The parameter settings in the experiments used for kNN and SVM are illustrated in Tables 2 and 3.

Table 2: kNN Parameters

Parameter	**Instances/Values**
k	k neighbors with $k = 5, 6, 7$
ω	Limit of boundary vectors stored in the system $\omega = 50$
$uOption = 1$	Majority vote *(MV)*
$uOption = 2$	Distance-based maximum vote *(DMV)*
$uOption = 3$	Two-thirds majority vote *(TMV)*
$uOption = 4$	Weighted vote *(WV)* with different thresholds $\varepsilon = 0.1, 0.2, 0.3, 0.4, 0.5$

Table 3: SVM Parameters

Parameter	**Instances/Values**
k in initPhase	$k = 6$
svm_update (su)	$su = 6, 12, 18$
ω	Limit of boundary vectors stored in the system $\omega = 50$
svm_threshold (st)	$st = 0.1, 0.2, 0.3, 0.4$ (for the uncertainty region)

In general, we distinguish two classes of parameters in Tables 2 and 3: those that are simple choices of options/methods for computing the boundary samples (uOption $1 \cdots 4$) and those that indicate a certain user-defined input. Given their number, the range of the least impacting parameters after initial empirical experiments, was squeezed, while those that clearly have effect on the outcome

were varied (k, su, st and thresholds in uOption 4).

In addition, both methods (kNN and SVM) have been examined with a predefined budget covering the following parameters based on the budget definition in Section 5, where $s = 100$ (i.e., budget window) and budget $B = 0.2$ (i.e., 20% of s). The results are visualized in diagrams and summarized in the following sections for the different classification methods and datasets.

6.3.1. kNN Results

The results of the experiments for kNN are discussed in the following. In particular, the effect of the uncertainty strategies (*uOptions*), the number of neighbors (k), and the influence of the budget are illustrated based on ER and Q.

Synthetic datasets. Figure 6 shows the results for SD, whereas Figure 7 depicts the results for the non-contiguous dataset NCSD. The first line of the figures shows ER and Q for all combinations of k and *uOption*. In addition, the second line shows the details for $uOption = 4$ for different values of k and ε.

Regarding Q, high numbers of Q are caused by the $uOption = 2$ (DMV). The lowest Q for SD and NCSD is reached with $uOption = 1$ (MV), but it has a worse ER. For SD, the $uOption = 3$ (TMV) performs slightly better than $uOption = 4$ (WV). Both options show a good compromise between Q and ER. For NCSD, the same is true for $uOption = 2$ (DMV) and $uOption = 4$ (WV). ER of the WV is small compared to the results of the other approaches (see ER of Figure 6(a)/Figure 7(a) and ER of Figure 6(c)/Figure 7(c)). The threshold $\varepsilon = \{0.1, 0.2\}$ results in a higher ER compared to the other threshold settings, but it has reasonably fewer queries Q. There is no obvious influence of k on ER. For the $uOption = 4$ a high value of k results in a higher number of Q.

The budget reduced the number of queries performed. For SD, the budget has a positive influence on the number of queries (e.g., compare Q of Figure 6(a) with Figure 6(b)). In addition, the ER remains almost constant (i.e., only less degradation, see ER of Figure 6(a) and Figure 6(b)). This is also true for the NCSD (see Figure 7).

Summarized for kNN and both synthetic datasets, the $uOption = 3$ and $uOption = 4$ show good performance. $uOption = 3$ is slightly better compared to $uOption = 4$. The $uOption = 2$ results in a high value of Q and $uOption = 1$ shows the smallest Q but the worst ER. Q can be reduced considerably by introducing the budget without a noticeable deterioration of ER.

Real-world datasets. Figure 8 portrays for CF the same experiments discussed so far. Again $uOption = 1$ shows the lowest number of queries. $uOption = 2$ has a high number of Q for all distances used independent of the budget (see Figure 8). Also, the $uOption = 4$ has high Q values, but a slightly smaller number compared to $uOption = 2$. In addition, small values of k (i.e., $k = 5$) show the best performance. The budget reduces the overall number of Q, independent of the metric used. For ER and Q, $uOption = 3$ (TMV) shows the best results.

The results of the AB dataset can be found in Figure 9. $uOption = 2$ also results in a high value of Q. $k = 5$ yields the best results. Compared to CF, indeed more queries per batch are needed. The $uOption = 4$ shows worse quality in the results compared to other datasets. This is as expected due to the characteristic of the dataset (e.g., high number of similarly items and differently labeled items). Only for a higher number of Q (i.e., due to a high threshold ε), the $uOption = 4$ shows good results. The introduction of a budget decreases in most cases Q, resulting only in a slight increase of ER for this dataset.
For AB and CF, small values of k yield better results. $uOption = 3$ (TMV) shows good results for AB and CF based on Q and ER. The TMV is less prone to differently labeled items, by just considering the majority of labels in contrast to the distance weighting in WV.

6.3.2. SVM Results

The SVM results for the different datasets are given in the following. The SVM evaluation parameters, i.e., update and threshold settings, are illustrated and discussed based on ER and Q.

Synthetic datasets. The diagrams of SVM for SD and NCSD are given in

Figure 10. The diagrams show the parameter settings for *svm_update* (*su*) and *svm_threshold* (*st*), respectively. The performance for SD and NCSD is very good. Regarding Q, the higher the threshold *st* for SVM, the higher Q. The introduction of the budget significantly reduces Q for SD (see Figure 10, first line); Q is cut in half. For NCSD, Q has not been reduced significantly. This is an indicator that Q for NCSD does not massively exceed the 20% budget marking. In general, SVM performs similarly to the $uOption = 4$ of kNN. For the synthetic datasets, SVM has a better overall performance considering Q resulting in a similar ER compared to kNN.

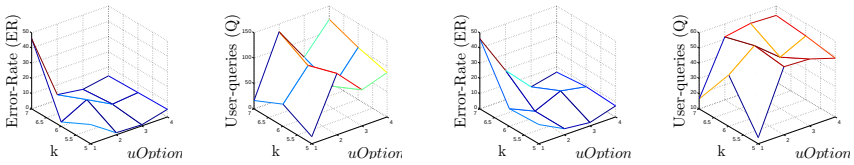
Real-world datasets. The results of SVM with CF and AB can be found in Figure 11. For CF and AB, only higher Q values (i.e., due to higher *st* values) show good performance, but indeed it is not better than kNN. This is due to the overlaps contained within the AB dataset (see Section 6.1.2). It is easier to distinguish this overlapping information based on the boundary neighborhood of the incoming item. For CF, SVM produces a better performance compared to kNN for higher ε.

6.4. Discussion

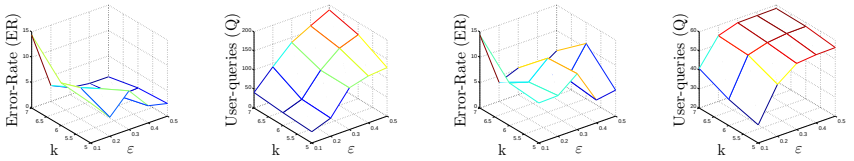
The proposed active classification approach, OBAL, builds upon calculated boundary vectors. The boundary vectors limit the number of items for calculating the k nearest neighbors. In addition, they also reduce the number of training items for the SVM. Concept drift when occurring is captured by the continuous calculation of the new boundary items from labeled data.

Table 4 summarizes the best results obtained from the experiments (i.e., with and without budget B) for each dataset. To identify the best results, we introduced a combined measure (see CM in Eq. 4), taking the average error-rate (ER) of the batches Bt and the number of queries (Q) based on the overall number of items (# items) into consideration.

$$CM = [0.8 * \frac{\sum_{i=1}^{|Bt|}(1 - ER_i/100)}{|Bt|}] + [0.2 * (1 - (Q/\#items))] \quad (4)$$

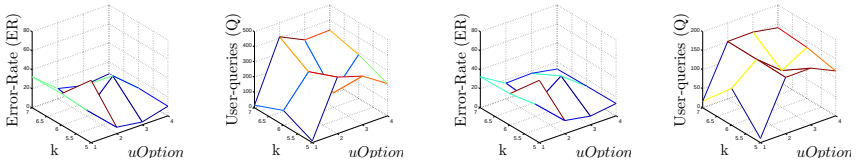


(a) $uOption = \{1, 2, 3\}$ without budget (b) $uOption = \{1, 2, 3\}$ with budget

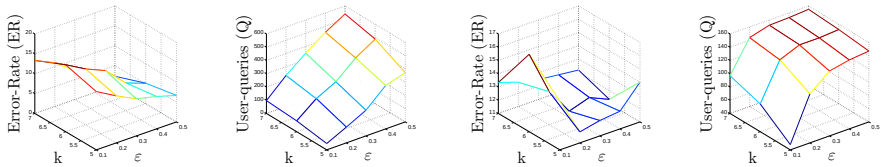


(c) $uOption = 4$ without budget (d) $uOption = 4$ with budget

Figure 6: kNN (SD): Performance on k and $uOption$ ($\omega = 50$)

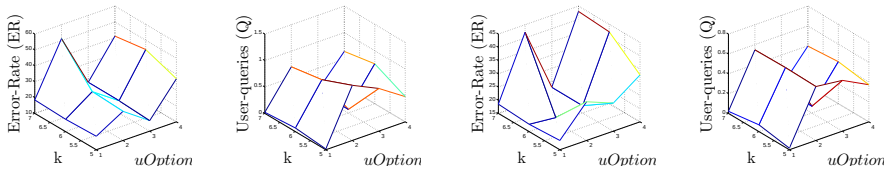


(a) $uOption = \{1, 2, 3\}$ without budget (b) $uOption = \{1, 2, 3\}$ with budget

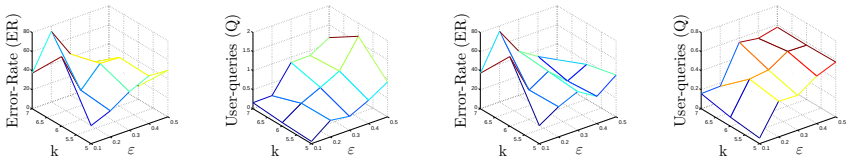


(c) $uOption = 4$ without budget (d) $uOption = 4$ with budget

Figure 7: kNN (NCSD): Performance on k and $uOption$ ($\omega = 50$)

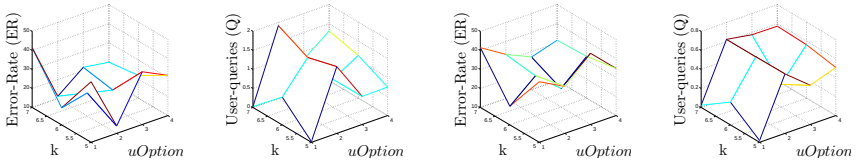


(a) $uOption = \{1, 2, 3\}$ without budget (b) $uOption = \{1, 2, 3\}$ with budget

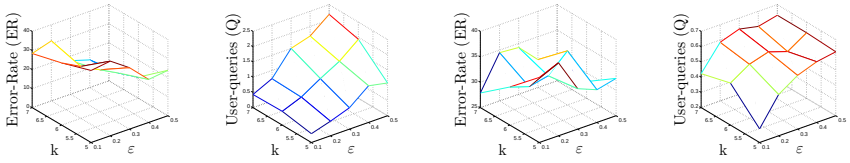


(c) $uOption = 4$ without budget (d) $uOption = 4$ with budget

Figure 8: kNN (CF): Performance on k and $uOption$ ($\omega = 50$)

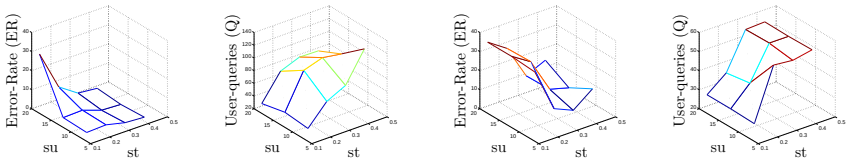


(a) $uOption = \{1, 2, 3\}$ without budget (b) $uOption = \{1, 2, 3\}$ with budget



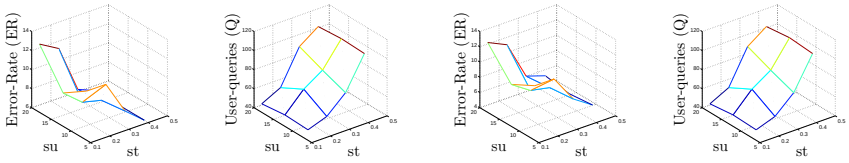
(c) $uOption = 4$ without budget (d) $uOption = 4$ with budget

Figure 9: kNN (AB): Performance on k and $uOption$ ($\omega = 50$)



(a) SD without budget

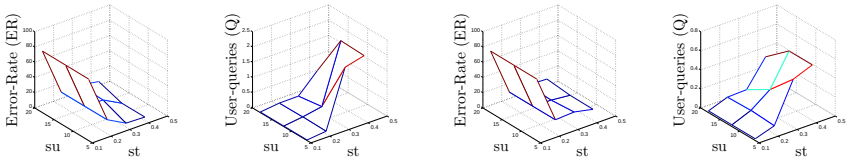
(b) SD with budget



(c) NCSD without budget

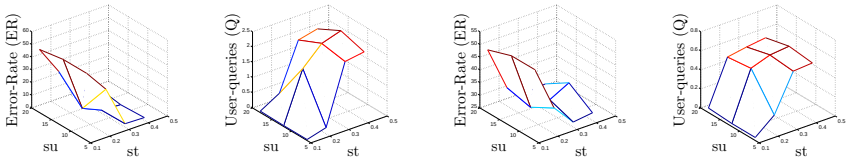
(d) NCSD with budget

Figure 10: SVM (SD, NCSD): Performance on *su* and *st* ($\omega = 50$)



(a) CF without budget

(b) CF with budget



(c) AB without budget

(d) AB with budget

Figure 11: SVM (CF, AB): Performance on *su* and *st* ($\omega = 50$)

Table 4: Best results of kNN and SVM (with and without budget)

kNN					
Dataset	uOption	k	Q (Q/#items)	ER	CM
SD	3	6	175.0 (0.22)	2.8750	0.9333
NCSD	3	5	320.0 (0.53)	3.0000	0.8693
CF	3	5	26.0 (0.03)	17.7791	0.8524
AB	3	5	275.0 (0.26)	18.9630	0.7961
kNN with budget					
Dataset	uOption	k	Q (Q/#items)	ER	CM
SD	3	5	216.0 (0.27)	2.2500	0.9280
NCSD	3	5	149.0 (0.25)	9.1667	0.8770
CF	3	5	26.0 (0.03)	17.7791	0.8524
AB	3	6	90.0 (0.09)	19.8134	0.8244
SVM					
Dataset	st	su	Q (Q/#items)	ER	CM
SD	0.1	6	161.0 (0.20)	4.6250	0.9227
NCSD	0.4	6	110.0 (0.18)	6.0000	0.9153
CF	0.2	18	35.0 (0.04)	18.2267	0.8470
AB	0.3	6	810.0 (0.77)	2.6834	0.8247
SVM with budget					
Dataset	st	su	Q (Q/#items)	ER	CM
SD	0.3	6	222.0 (0.28)	9.2500	0.8705
NCSD	0.4	12	112.0 (0.19)	5.6667	0.9173
CF	0.2	18	35.0 (0.04)	18.2267	0.8470
AB	0.3	6	221.0 (0.21)	27.0141	0.7419

In a nutshell, independently of the datasets, we can observe the following:

- The *distance-based majority vote (DMV)* uncertainty strategy (i.e., $uOption = 2$) of the kNN-based approach needs on average more queries compared to other uncertainty strategies used with kNN.

- The kNN-based $uOption = 1$, *maximum vote (MV)*, produces the worst results. The number of queries is very low but the error-rate (ER) is high.

- By introducing the budget, the number of queries (Q) decreases considerably causing only a marginally negative effect on ER compared to the

settings without budget (compare Fig. 6(a) with Fig 6(b)).

- For kNN, $k = 5$ is preferred, especially due to the lower number of queries (Q). For SVM, a higher update rate (i.e., $su = 6$) is recommended following the results in this study.

- Regardless of the dataset used, the two-thirds majority vote uncertainty strategy ($uOption = 3$) is better than other uncertainty strategies.

- kNN shows good behavior in the case of overlapping data and the SVM has an overall good performance considering Q.

In terms of the effect of the dataset on the performance of OBAL, the following remarks can be made:

- In the case of synthetic data (SD and NCSD), the average number of queries required by SVM to distinguish between relevant and irrelevant items is much less lower than that required by kNN (see Figs. 6, 7, and 10).

- For Australian Bushfires (AB), the number of queries required by SVM is much higher due to the largest mismatch with the ground truth.

- In the case of Colorado Floods (CF), the number of queries for SVM is in general similar or slightly smaller compared to kNN, see for example Figures 8 and 11.

- Higher number of neighbours, k, leads to higher number of queries, Q.

- Lower values of ε results in lower number of queries but higher error. It is therefore recommended to make a trade-off between the querying budget and classification quality.

- Intuitively higher budget leads to better classification quality since the number of labelled examples increases yielding a better insight into the distribution of data from different classes.

- kNN produces overall the best results as shown in Table 4.

In terms of robustness and as outlined above, the sensitivity analysis shows quite informative indications. Looking at Fig. 6 to 11 the behaviour of active learning using either kNN or SVM is quite regular across all datasets (both synthetic and real-world), which indicates that for any combination of parameters OBAL behaves very similarly across all datasets. Any good solution tends to be consistently good.

Finally, OBAL was driven by the application of crisis management, and in this study two datasets were used: Colorado Floods (CF) and Australian Bushfires (AB) datasets. The use of budget for the best option did not have much impact in the case of CF, while for AB the use of budget was influential. Indeed AB is more challenging as described in Sec. 6.1.2 and OBAL did react to budget. One has to make the tradeoff between the budget and the quality of the classification. KNN seems to do better than SVM in terms of tradeoff. Overall, OBAL fits the scenario where classes overlap over time in the separating region and some guidance is required to learn in an online way that boundary.

6.5. Unbalanced Data

It could be the case that social media data about a crisis is unbalanced. For our experiments, we used real-world datasets as benchmarks. Within these datasets, the unbalanced problem is not acute (see Table 5). Furthermore, we observed that the Australian Bushfire (AB) dataset, which is quite balanced, turned out to be the most challenging dataset (see Section 6.4).

Table 5: Unbalance ratio of the real-world datasets

Dataset	**Relevant**	**Irrelevant**	**Ratio**
Australian Bushfires (AB)	645	408	1.5
Colorado Floods (CF)	751	224	3.3

In the context of crisis management, it is necessary to flag out positive examples (i.e., relevant items). Even, if the system predicts a non-relevant item

as relevant, there is no substantial cost for decision making, except that the crisis management person (first responder) will probably receive some irrelevant tweets.

In this paper, we did not investigate the aspect of crisis data unbalance, but it is a relevant and important issue to be fully addressed in the future.

7. Conclusion

This paper presents a batch-based active learning approach used for a binary classification problem to distinguish between relevant and irrelevant information contained in a data stream. The research can be applied to other fields as well. Especially to areas were changes in data happens and thus continuous learning is needed (e.g. spam detection in social networks). The OBAL algorithm relies on the uncertainty criterion to query the data samples. In addition, several uncertainty strategies for requesting feedback from the user have been implemented and tested. OBAL was extensively evaluated using two classifiers kNN and SVM, different uncertainty strategies, different parameter settings, and two classes of datasets: synthetic and real-world datasets.

The proposed algorithm is dedicated to support the crisis management team to identify relevant and useful information communicated by people (citizen journalists) in social media in order to organize the rescue and intervention plans. Because automatic classification requires labeled items, it is extremely difficult to have that information in real-time as it requires human-in-the-loop to do that. Active learning is an adequate methodology to enable labelling of items (Tweets) to be used by the classifier. The challenge in active learning is to label as few items as possible, while keeping the quality of the prediction high. Several criteria can be used and here we focused on a number of label-uncertainty strategies. OBAL is designed to actively query streaming items whose classes are undecidable (falling in the region separating relevant and irrelevant classes) to be used for further learning by the classifier.

Experiments show a very good performance; an error rate (between 0.02 and

0.27) in most cases is obtained. In the future, we will investigate other active learning criteria, like the density-based criterion and we will combine them to cope with drifting data streams, where drift can happen either at the separating region between classes or far from the known boundaries (called remote drift).

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 261817 and was partly performed in the Lakeside Labs research cluster at Alpen-Adria-Universität Klagenfurt. A. Bouchachia was partly supported by the European Commission under the Horizon 2020 Grant 687691 related to the Project: PROTEUS: Scalable Online Machine Learning for Predictive Analytics and Real-Time Interactive Visualization.

References

Abel, F., Hauff, C., Houben, G.-J., Stronkman, R., & Tao, K. (2012). Twitcident: Fighting Fire with Information from Social Web Streams. In *Proc. of the 21st International Conference companion on World Wide Web* WWW '12 Companion (pp. 305–308). New York, NY, USA: ACM.

Attenberg, J., & Provost, F. (2011). Online Active Inference and Learning. In *Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* KDD '11 (pp. 186–194). NY, USA: ACM.

Borchani, H., Martínez, A. M., Masegosa, A. R., Langseth, H., Nielsen, T. D., Salmerón, A., Fernández, A., Madsen, A. L., & Sáez, R. (2015). Modeling concept drift: A probabilistic graphical model based approach. In E. Fromont, T. De Bie, & M. van Leeuwen (Eds.), *Advances in Intelligent Data Analysis XIV: 14th International Symposium, IDA 2015, Saint Etienne. France, October 22 -24, 2015. Proceedings* (pp. 72–83). Cham: Springer International Publishing.

Bouchachia, A. (2014). *Regularised Semi-Supervised Learning*. Internal report 1 Bournemouth University, UK.

Boyd, D., Golder, S., & Lotan, G. (2010). Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter. In *Hawaii International Conference on System Sciences (HICSS)* (pp. 1–10).

Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006). Worst-Case Analysis of Selective Sampling for Linear Classification. *J. Mach. Learn. Res.*, *7*, 1205–1230.

Chatzilari, E., Nikolopoulos, S., Kompatsiaris, Y., & Kittler, J. (2014). Active Learning in Social Context for Image Classification. *9th International Conference on Computer Vision Theory and Applications, VISAPP*, .

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine learning*, *15*, 201–221.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *arXiv preprint cs/9603104*, .

da Costa, F., Rios, R., & de Mello, R. (2016). Using dynamical systems tools to detect concept drift in data streams. *Expert Systems with Applications*, *60*, 39 – 50.

Dasgupta, S., & Hsu, D. (2008). Hierarchical Sampling for Active Learning. In *Proc. of the 25th International Conference on Machine Learning* ICML '08 (pp. 208–215). New York, NY, USA: ACM.

Denef, S., Bayerl, P. S., & Kaptein, N. (2013). Social Media and the Police - Tweeting Practices of British Police Forces during the August 2011 Riots. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI13)*. Paris, France.

Dudani, S. A. (1976). The Distance-Weighted k-Nearest-Neighbor Rule. *IEEE Trans. on Systems, Man and Cybernetics*, *SMC-6*, 325–327.

Gama, J. a., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv.*, *46*, 44:1–44:37.

Guo, L., & Liew, A. W. C. (2016). Online-offline extreme learning machine with concept drift tracking for time series data. In *International Conference on Digital Image Computing: Techniques and Applications (DICTA)* (pp. 1–6).

Hughes, A., Denis, L. S., Palen, L., & Anderson, K. (2014). Online Public Communications by Police & Fire Services during the 2012 Hurricane Sandy. In *Proc. of the ACM 2014 Conference on Human Factors in Computing Systems (CHI)*. Toronto.

Ienco, D., Bifet, A., Žliobaitė, I., & Pfahringer, B. (2013). Clustering Based Active Learning for Evolving Data Streams. In J. Frnkranz, E. Hüllermeier, & T. Higuchi (Eds.), *Discovery Science* (pp. 79–93). Springer Berlin Heidelberg volume 8140 of *Lecture Notes in Computer Science*.

Ienco, D., Pfahringer, B., & Zliobaite, I. (2014). High Density-Focused Uncertainty Sampling for Active Learning over Evolving Stream Data. In *International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications* (pp. 133–148).

Imran, M., Castillo, C., Lucas, J., Meier, P., & Rogstadius, J. (2014a). Coordinating Human and Machine Intelligence to Classify Microblog Communications in Crises. In *11th International Conference on Information Systems for Crisis Response and Management*.

Imran, M., Castillo, C., Lucas, J., Meier, P., & Vieweg, S. (2014b). AIDR: Artificial Intelligence for Disaster Response. In *Proc. of the Companion Publication of the 23rd International Conference on World Wide Web Companion* WWW Companion '14 (pp. 159–162). Republic and Canton of Geneva, Switzerland: International World Wide Web Conference Steering Committee.

Jin, X., Lin, C., Luo, J., & Han, J. (2011). A Data Mining-based Spam Detection

System for Social Media Networks. *Proc. of the 37th Very Large Data Bases (VLDB) Endowment*, *4*.

Klein, B., Laiseca, X., Casado-Mansilla, D., Lpez-de Ipia, D., & Nespral, A. (2012). Detection and Extracting of Emergency Knowledge from Twitter Streams. In J. Bravo, D. Lpez-de Ipia, & F. Moya (Eds.), *Ubiquitous Computing and Ambient Intelligence* Lecture Notes in Computer Science (pp. 462–469). Springer Berlin Heidelberg.

Kumar, S., Morstatter, F., Zafarani, R., & Liu, H. (2013). Whom Should I Follow? Identifying Relevant Users During Crises. In *Proc. of the ACM Conference on Hypertext and Social Media*. Paris, France.

Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In *Soviet physics doklady* (p. 707). volume 10.

Li, C.-T., Shan, M.-K., Jheng, S.-H., & Chou, K.-C. (2016). Exploiting concept drift to predict popularity of social multimedia in microblogs. *Information Sciences*, *339*, 310 – 331.

Li, R., Lei, K. H., Khadiwala, R., & Chang, K.-C. (2012). TEDAS: A Twitter-based Event Detection and Analysis System. In *IEEE 28th International Conference on Data Engineering (ICDE)* (pp. 1273–1276). doi:10.1109/ICDE.2012.125.

MacEachren, A., Jaiswal, A., Robinson, A., Pezanowski, S., Savelyev, A., Mitra, P., Zhang, X., & Blanford, J. (2011). SensePlace2: GeoTwitter Analytics Support for Situational Awareness. In *IEEE Conference on Visual Analytics Science and Technology (VAST)* (pp. 181 –190).

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Maxwell, D., Raue, S., Azzopardi, L., Johnson, C., & Oates, S. (2012). Crisees: Real-Time Monitoring of Social Media Streams to Support Crisis Manage-

ment. In R. Baeza-Yates, A. Vries, H. Zaragoza, B. Cambazoglu, V. Murdock, R. Lempel, & F. Silvestri (Eds.), *Advances in Information Retrieval* (pp. 573–575). Springer Berlin Heidelberg volume 7224 of *Lecture Notes in Computer Science*.

Mohamad, S., Bouchachia, A., & Sayed-Mouchaweh, M. (2017). A bi-criteria active learning algorithm for dynamic data streams. *IEEE Transactions on Neural Networks and Learning Systems*, *PP*, 1–13.

Morstatter, F., Kumar, S., Liu, H., & Maciejewski, R. (2013). Understanding Twitter Data with TweetXplorer. In *Proc. of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* KDD '13 (pp. 1482–1485). New York, NY, USA: ACM.

Nguyen, H. T., & Smeulders, A. (2004). Active learning using pre-clustering. In *Proceedings of the 21st International Conference on Machine learning* (p. 79). ACM.

Olteanu, A., Vieweg, S., & Castillo, C. (2015). What to Expect When the Unexpected Happens: Social Media Communications Across Crises. *In Proc. of the ACM 2015 Conference on Computer Supported Cooperative Work and Social Computing (CSCW)*, .

Papadopoulos, S., Schinas, E., Mironidis, T., Iliakopoulou, K., Spyromitros-Xioufis, E., Tsampoulatidis, I., & Kompatsiaris, Y. (2013). Social Multimedia Crawling and Search. URL: `http://stcsn.ieee.net/e-letter/vol-1-no-3/social-multimedia-crawling-and-search`.

Petkos, G., Papadopoulos, S., Schinas, E., & Kompatsiaris, Y. (2014). Graph-Based Multimodal Clustering for Social Event Detection in Large Collections of Images. In C. Gurrin, F. Hopfgartner, W. Hurst, H. Johansen, H. Lee, & N. O'Connor (Eds.), *MultiMedia Modeling* (pp. 146–158). Springer International Publishing volume 8325 of *Lecture Notes in Computer Science*.

Pohl, D. (2014). Social Media Analysis for Crisis Management: A Brief Survey. *IEEE Computer Society Special Technical Community on Social Networking E-Letter*, *2*.

Pohl, D., Bouchachia, A., & Hellwagner, H. (2012). Automatic Sub-Event Detection in Emergency Management Using Social Media. In *In First Inter. Workshop on Social Web for Disaster Management (SWDM), In conjunction with WWW'12*. Lyon, France.

Pohl, D., Bouchachia, A., & Hellwagner, H. (2013). Social Media for Crisis Management: Clustering Approaches for Sub-Event Detection. *Multimedia Tools and Applications*, (pp. 1–32).

Pohl, D., Bouchachia, A., & Hellwagner, H. (2015). Online Indexing and Clustering of Social Media Data for Emergency Management. *Neurocomputing*, *172*, 168–179. Special Issue on Social Media Analytics and Learning.

Pozzolo, A. D., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015). Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8).

Settles, B. (2010). Active Learning Literature Survey. *University of Wisconsin, Madison*, *52*, 55–66.

Sheth, A., Purohit, H., Jadhav, a., Kapanipathi, P., & Chen, L. (2011). Understanding Events Through Analysis of Social Media. In *Proc. of the 20th International World Wide Web Conference (WWW)*. Hyderabad, India.

Smailović, J., Grčar, M., Lavrač, N., & Žnidaršič, M. (2014). Stream-based Active Learning for Sentiment Analysis in the Financial Domain (in press). *Information Sciences*, .

Song, G., Ye, Y., Zhang, H., Xu, X., Lau, R. Y., & Liu, F. (2016). Dynamic clustering forest: An ensemble framework to efficiently classify textual data stream with concept drift. *Information Sciences*, *357*, 125 – 143.

Starbird, K. (2011). Digital Volunteerism During Disaster: Crowdsourcing Information Processing. In *Conference on Human Factors in Computing Systems*.

Vijayanarasimhan, S., Jain, P., & Grauman, K. (2010). Far-Sighted Active Learning on a Budget for Image and Video Recognition. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3035–3042).

Žliobaitė, I., Bifet, A., Pfahringer, B., & Holmes, G. (2014). Active Learning With Drifting Streaming Data. *IEEE Trans. on Neural Networks and Learning Systems*, *25*, 27–39.

Webb, R., Geoffrey I.and Hyde, Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, *30*, 964–994.

Yin, J., Lampert, A., Cameron, M., Robinson, B., & Power, R. (2012). Using Social Media to Enhance Emergency Situation Awareness. *Intelligent Systems, IEEE*, *27*, 52–59.

Zhou, Y., Yang, L., Van de Walle, B., & Han, C. (2013). Classification of Microblogs for Support Emergency Responses: Case Study Yushu Earthquake in China. In *46th Hawaii International Conference on System Sciences (HICSS)* (pp. 1553–1562). doi:10.1109/HICSS.2013.129.

Zhu, X., Zhang, P., Lin, X., & Shi, Y. (2007). Active Learning from Data Streams. In *Seventh IEEE International Conference on Data Mining* (pp. 757–762).

Zhuang, J., Mei, T., Hoi, S. C., Hua, X.-S., & Li, S. (2011). Modeling Social Strength in Social Media Community via Kernel-based Learning. In *Proc. of the 19th ACM International Conference on Multimedia* MM '11 (pp. 113–122). New York, NY, USA: ACM.

Algorithm 1 : OBAL algorithm

Input: stream of data, parameters ω, ζ such that ($\zeta \leq \omega$), b_t=0, budget B

For an unlabeled input $\mathbf{x}$
if ($|X| < \omega$) *or* ($|Boundary| < \zeta$) **then**
  Query the label l of $\mathbf{x}$ and set $b_t = b_t + 1$
  $X_l = X_l \cup \{\mathbf{x}\}$
  if $|X| \geq 2$ **then**
    Find the subset of boundary examples, *Boundary*, in the labeled set X using Alg. 2
    if $|X| > \omega$ **then**
      X=*Boundary*
    end if
  end if
else
  uRes = Classify(Boundary, $\mathbf{x}$)
  if uRes.uncertainty = true **then**
    if the number of queries already placed, b_t, is less than a threshold, B (See Sec. 5) **then**
      Query the label l of $\mathbf{x}$
      $X_l = X_l \cup \{\mathbf{x}\}$
      Find the subset of boundary examples in the labeled set X, *Boundary*, using Alg. 2
      if $|X| > \omega$ **then**
        X=*Boundary*
      end if
    else
      Assert *uRes.label* as label for $\mathbf{x}$
    end if
  else
    Assert *uRes.label* as label for $\mathbf{x}$
  end if
end if

Algorithm 2 : ComputeBoundary ()

Input: X, distance-measure and C (number of class labels)

$Bd = C \times C$
for pair of classes (i,j) **do**
  Compute distance matrix $D^{i,j}$ between X^i and X^j based on *distance-measure*
  for p = 1 to $|X^i|$ **do**
    for q = 1 to $|X^j|$ **do**
      $m = \frac{X^i(p)+X^j(q)}{2}$
      $T = X^i \cup X^j$
      Compute distance $Z(l)$ using *distance-measure* between each data sample in $T(l)$ and m where $l = 1 \ldots |T|$
      Find the set R of data samples l for which $Z(l) < D^{i,j}(p,q)/2$
      if $R = \emptyset$ **then**
        $Bd(i,j) = Bd(i,j) \cup p$
        $Bd(j,i) = Bd(j,i) \cup q$
      end if
    end for
  end for
end for
return Bd

Algorithm 3 : Classify_kNN

Input: Boundary, k and **x**

kNN-result = kNN(Boundary, k)
uRes = isUncertainty(kNN-result, Boundary)

Algorithm 4 : Classify_SVM

Input: Boundary and input **x**

1: **if** ($svm\_start == 1$) *and* $svm\_update == 0$) **then**
2: svm = SVMTrain(Boundary)
3: svm_start = 0
4: **end if**
5: result = SVMClassify(**x**)
6: uRes = isUncertaintySVM(result,**x**)
7: **if** uRes.uncertainty == true **then**
8: $uRes.assigned\_label$ = svm-result {//add default label}
9: **end if**

Algorithm 5 : updateBoundary(X, Boundary)

Input: X and $Boundary$

1: **if** $|X| > \omega$ **then**
2: X=$Boundary$
3: **end if**
4: **for** i **to** C **do**
5: remove oldest boundary vectors of class i in $Boundary$ with $|Boundary_i| > \omega$
6: **end for**
7: **return** X and $Boundary$

Algorithm 6 : isUncertainty_MV(res, Boundary)

Input: kNN-result res and boundaries $Boundary$

1: $Count_i$ = number of neighbors in $res.neighbours$ belonging to class i
2: $max\_classes = \arg\max_{i=1\cdots c} Count_i$
3: **if** $|max\_classes| > 1$ **then**
4: $uRes.uncertainty = true$
5: **else**
6: $uRes.uncertainty = false$
7: $uRes.assigned\_label = max\_classes$
8: **end if**
9: **return** uRes

Algorithm 7 : isUncertainty_DMV(res, Boundary)

Input: kNN-result res and boundaries $Boundary$

$Count_i$ = number of neighbors in $res.neighbours$ belonging to class i

$max\_classes = \arg\max_{i=1\cdots c} Count_i$

$aDist$ = average distance in $res.neighbors\_distance$ to all neighbors

$count\_distance$ = number of neighbors for which $res.neighbors\_distance < aDist$

if $(|max\_classes| > 1)$ or $(count\_distance < (|res.neighbors|/2))$ **then**

  $uRes.uncertainty = true$

else

  $uRes.uncertainty = false$

  $uRes.assigned\_label = max\_classes$

end if

return uRes

Algorithm 8 : isUncertainty_TMV(res, Boundary)

Input: kNN-result res and boundaries $Boundary$

Calculate majority: $majority = (|res.neighbors| * 2/3)$

$Count_i$ = number of neighbors in $res.neighbours$ belonging to class i

$max\_classes = \arg\max_{i=1\cdots c} Count_i$

$max\_count = max_{i=1\cdots c} Count_i$

if $|max\_count| >= majority$ **then**

  $uRes.uncertainty = false$

  $uRes.assigned\_label = max\_classes$

else

  $uRes.uncertainty = true$

end if

return uRes

Algorithm 9 : isUncertainty_WV(res, Boundary)

Input: kNN-result *res* and boundaries *Boundary*; ε threshold

1: $d_{max} = \max(res.neighbors\_distance)$

2: $d_{min} = \min(res.neighbors\_distance)$

3: $value_l$ is the sum of weighted distances w_k calculated for all k neighbors with distance d_k belonging to class l (see also Dudani (1976)):

$$w_k = \begin{cases} \frac{d_{max}-d_k}{d_{max}-d_{min}} & if\ d_{max} \neq d_{min} \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

4: calculate difference $value_{l,k} = abs(value_l - value_k)$ between all pairs of classes l and k

5: **if** $mean(value_{l,k}) < \varepsilon$ **then**

6: $\quad uRes.uncertainty = true$

7: **else**

8: $\quad uRes.uncertainty = false$

9: $\quad uRes.assigned\_label = \arg\max_{l=1\dots c} value_l$

10: **end if**

11: **return** uRes

Algorithm 10 : isUncertaintySVM(res, input)

Input: SVM-result *res*, current *input*

1: $U = svm.supportVectors$

2: $svm\_distance = \sum_{u_i \in U} \alpha_i K(u_i, input) + b$ with b is the bias, α_i are the weights, and rbf-kernel $K(x_1, x_2) = e^{(-\eta(||x_1 - x_2||^2))}$ with $\eta = 1/2$

3: **if** $svm\_distance < svm\_threshold$ **then**

4: $\quad uRes.uncertainty = true$

5: **else**

6: $\quad uRes.uncertainty = false$

7: $\quad uRes.assigned\_label = res.label$

8: **end if**

9: **return** uRes