Time-aware online reputation analysis
Peetz, M.H.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Active Learning for Filtering of Streaming Documents

With increasing volumes of social media data, monitoring and analyzing this data is a vital part of the marketing strategy of businesses. The extraction of topics, reputation, and trends around an entity (such as a company, organization, celebrity) allows analysts to understand and manage the entity’s reputation. It is no longer feasible to manually process every tweet or blogpost that may have been written about an entity and reputation analysts would like to have this step automated (see Chapter 4). Since entity names are often ambiguous [279], filtering social media for relevant information—that is, entity filtering (EF)—saves tedious work and is a vital pre-processing step for further automation of online reputation management (ORM) [230]. However, if the performance of the EF module decreases, the performance of all subsequent modules is harmed [230]. Automatic EF on social media is therefore an active field of research and has previously been considered in various settings: at the WePS-3 evaluation effort [5] and as part of the RepLab 2012 and 2013 challenges [6, 7].

Missing important tweets and news items about an entity of interest can potentially be disastrous and expensive: when users on Twitter found out about H&M deliberately destroying perfectly wearable winter jackets, this incident went viral and caused bad publicity [195]. Communications experts around an entity may need to react immediately to avoid long-lasting harmful publicity. The ORM industry therefore seeks to find a balance between manual and automatic filtering. Currently, monitoring platforms like Topsy\(^1\) or dashboards at HootSuite\(^2\) allow for keyword filtering. For keyword filtering analysts have a list of keywords collected over time and they reuse them every day. This approach leads to high recall, but not necessarily to high precision. Hence, reputation analysts still need to inspect many non-relevant tweets. However, keywords used for filtering can be customized for precision, but as a consequence, new topics with critical tweets may not reach the analysts. In active learning, the learner samples instances, tweets, that should be annotated manually. Those annotations then feedback into the model. This sampling can be informed or random. Active learning is especially attractive in this setting of EF for ORM because it promises to (a) use the analysts’ background knowledge and understanding to improve an EF system, and (b) capture new topics and

\(^1\)http://topsy.com
\(^2\)http://hootsuite.com
problems without exhaustive annotation efforts.

Topics and events surrounding entities change over time within the stream of documents. By adopting a batch scenario, the RepLab 2013 set-up makes a number of simplifying assumptions. Spina [229] shows the viability of active learning for an active learning adjusted batch scenario of RepLab 2013. However, a batch scenario in a real life ORA scenario would correspond to doing active learning on a static dataset. With new tweets constantly being posted and information changing over time, a streaming scenario is closer to the daily workflow of social media analysts. We investigate how active learning can help address the EF task in a streaming scenario. Based on the RepLab 2013 data, in this chapter we propose a new streaming scenario, capturing the changes of entity models over time. Firstly, we want to know if active learning is also viable for the streaming scenario:

**RQ5.1** For the entity filtering task, Does margin sampling improve effectiveness over random sampling, i.e., is it a strong baseline?

Active learning significantly outperforms passive learning in the streaming scenario as well. Here too, we find that margin sampling improves effectiveness over random sampling. Streaming microblog data calls for methods using temporal information. On the one hand, recent tweets may be more important to estimate a model for the future than older tweets. On the other hand, tweets published in bursts could give a better estimate of the topics that are important for an entity. Based on work in Chapter 7 for recency and Chapter 6 for bursts, we propose two new angles for sampling: based on recency and on bursts, respectively. Incorporating recency priors into margin sampling, we ask:

**RQ5.2** Does sampling based on recency priors and margin sampling together, outperform margin sampling with respect to F-score?

We also propose a temporal reranking, this can be based on bursts or the recency of publication:

**RQ5.3** Does temporal reranking of margin sampled results based on bursts or recency, outperform margin sampling with respect to F-score?

For both questions, we analyse the influence of a strong initial training model and we find that the impact on effectiveness of a large, bulk training set is strong, in particular for margin sampling. We show that the influence of bulk training in the streaming setting helps to build a strong initial model where not only the passive learner performs well, but also the informed selection of tweets for active annotations benefits strongly. We show that the effectiveness is higher for many entities using temporal approaches, in particular burst-based reranking of margin sampled candidate sets proves to be a promising sampling method.

Our contributions are a streaming entity filtering scenario closer to the real-life problem. We also contribute temporal sampling methods for active learning that perform well on temporally sensitive entities.

---

3These simplifications are dictated by the limitations of ensuring suitable experimental conditions for a community-based benchmark. Participants have to run their systems and submit runs that are subsequently evaluated by the organizers.
The chapter is organized as follows. We continue with a brief introduction to active learning in Section 8.1. We introduce our approaches to EF in Section 8.2. We proceed with an explanation of our experimental setup (Section 8.3) and analyse the results in Section 8.4. We conclude in Section 8.5.

### 8.1 Active Learning

Active learning [222] is a subfield of machine learning that is increasingly gaining interest. Unlike passive supervised learning, where the goal of a learner is to infer a classification model from labeled and static training data, active learning interacts with the user for updating the classifier. This learning framework has been widely used in information access tasks [218, 277] and, in particular, in text categorization [110, 149, 220, 225, 276]. As in passive text categorization, Support Vector Machines have proven to be one of the most competitive learning models in active learning [149, 218, 276].

So far, little work has been done applying active learning in streaming scenarios [54, 220, 261, 286]. A common approach to deal with streaming data, consists in dividing it into chunks. In [286], for each previous data chunk a base classifier is learned. Then, classifiers are combined to form a classifier committee, that is used to label the new chunk. Ensemble learning has also been used by [30, 179, 287]. A common finding is that in a classifier ensemble, variance corresponds to error rate [287]. In a non-streaming scenario, several approaches have been used to deal with the problem that documents where the classifier is uncertain, are not close to any other documents [113, 285]. Zhu et al. [285] introduce a density measure that they combine with an entropy-based uncertainty measure. Alternatively, they use the density measure to rerank the top uncertain documents. Ienco et al. [113] cluster the current new batch. They use information like the homogeneity of clusters to update the ranking of the samples. The assumptions behind this method are the same as for the density measure from [285]. We combine the sampled data of previously seen data to retrain a single classifier at each step. For selecting data to be annotated manually during the active learning, Žliobaitė et al. [261] compare random and fixed uncertainty margin sampling—equivalent to the margin sampling considered in our work—to other sampling methods that take into account the budget available to request feedback at each time. They alert at concept drift and then relabel. Chu et al. [54] minimize class bias issues with streaming data for Bayesian Learning. Their use of the decay factor for old samples is closest to the idea of weighting recent samples higher.

Apart from the difference in task, our work differs from previous work in that (1) we do not use external data, only tweets are considered to learn a model, and (2) new labeled instances are directly added to the training set used to update the model. The static scenario is described in Spina [229].

### 8.2 An Active Learning Approach to Entity Filtering

Our approach to entity filtering is based on active learning, a semi-automatic machine learning process interacting with the user for updating the classification model. It selects instances that are meant to maximize the classification performance with minimal effort.
Algorithm 3: Active learning for entity filtering (EF)

1. **Initialization**;
   - **Data**: Training dataset
2. **begin**
3. Initialize model;
4. **return** initialized model
5. **end**
6. **Training phase**;
   - **input**: Current model
   - **Data**: Training dataset
7. **begin**
8. Represent features;
9. Retrain model;
10. **return** (re)trained model
11. **end**
12. **Test phase**;
   - **Data**: Test dataset
13. Represent features;
14. **repeat**
15. Run current model on test data;
16. Select candidate samples for feedback;
17. Collect feedback;
18. Update training and test datasets;
19. Run training phase with updated training dataset;
20. **until** a suitable termination condition;

Algorithm 3 sketches the main steps of our active learning approach to entity filtering. First, the instances are represented as feature vectors. Second, the instances from the training dataset are used for building the initial classification model. Third, the test instances are automatically classified using the initial model. Fourth, we sample candidates to be offered to the user for additional labeling; this step is performed by margin sampling: the instance closest to the class separation is selected. Fifth, the user manually inspects the instance and labels it. The labeled instance is then considered when updating the model. The active learning process is repeated until a termination condition is satisfied.

We use a Support Vector Machine\(^4\) (SVM) classifier. Our active learning approach can be split into the selection of candidates for active annotations, annotation of the candidates and updating the model. Therefore, one iteration of our learning model follows the following three steps:

1. Select the best candidate \(x\) from the test set \(T\) (line 16 in Algorithm 3)
2. Annotate the candidate \(x\) (line 17 in Algorithm 3), and

3. Update the model (line 19 in Algorithm 3).

If the resources are available, the training data used to initialize the model can be a large manually annotated (bulk) set of tweets published before the test set. If this training set is available, we call this a warm start. Without a warm start, we have a cold start, where the initial model selects and classifies tweets randomly; the bulk set of training data facilitates a strong initial model. Below we detail the candidate selection, candidate annotation, and model updating in Sections 8.2.1, 8.2.2, and 8.2.3, respectively.

8.2.1 Candidate selection

Candidate selection is the process of sampling the candidates that are used for annotation. A successful selection approach selects candidates which, when annotated, improve the model. Standard baseline approaches are: passive learning without sampling which is identical to non-active learning, random sampling which samples randomly from the pool, and margin sampling which samples close to the margin of the classification boundary. We also propose two further approaches to improve margin sampling. For one, we add a recency prior, introduced in Chapter 8, to the sampling score. For reranking, we rerank the list of samples based on either bursts or recency. The oracle sampling denotes an upper bound for our sampling approaches.

Passive learning

Passive learning does not use any active learning at all. If we look at Algorithm 3, we only initialise the model without retraining it: We skip the Training phase and the Test phase.

Random sampling

For random sampling, the candidate instance is sampled without replacement from the training set. There is no informed prior on the instances. Random sampling has proven to be effective for other tasks, e.g., building dependency treebanks [13], or clinical text classification [80].

Margin sampling

The most commonly used sampling method in binary classification problems is uncertainty sampling [222]. We consider a specific uncertainty sampling method especially suitable for support vector machines [242]: margin sampling. We measure the uncertainty of a candidate $x$ based on the distance to the margin, so

$$\text{Uncertainty}(x) = 1 - |P(C_1 \mid F_x) - P(C_2 \mid F_x)|,$$  \hspace{1cm} (8.1)

where $P(C_1 \mid F_x)$ and $P(C_2 \mid F_x)$ are the probabilities that the candidate $x$, as represented by the feature vector $F_x$, generates the classes $C_1$ and $C_2$, respectively.

Candidates are sampled based on the classification difficulty, thereby selecting candidates where the classifier is less confident. Following this, the candidate $x$ to be annotated
from the test set $T$ is selected as follows:

$$x = \arg \max_{x_i \in T} \text{Uncertainty}(x_i).$$

This candidate $x$ is then annotated and used to update the model. For a linear kernel of the SVM this means: instances (tweets here) that are closest to the class separation are selected.

**Recency Prior**

We introduced the concept of a recency prior in Chapter 7. The intuition behind using a recency prior is that very recently published documents are more likely to have information that may improve filtering of the future. We select the candidate $x$ that:

$$x = \arg \max_{x_i \in T} \text{Uncertainty}(x_i) \cdot f(x_i, \max(T), g),$$

where $f(x_i, \text{latest}(T), g)$ is a retention function introduced in Section 7.2.2 in Chapter 7. The parameters for the retention function are the granularity $g$ and the latest sample $\text{latest}(T)$ in the set $T$. Chapter 7 showed that the Weibull function meets the requirements we set for priors better. Hence, we use $f_{\text{basic Weibull}}$ (see Section 7.6).

**Reranking**

We present two approaches to reranking: temporal and burst-based reranking. At the TREC-microblog 2011 [177] the ranked result list for the retrieval task was cut-off at a certain limit and reranked according to tweet id, an estimator for the arrival time. While this was for the evaluation of a retrieval task, this is very close to our reranking approach. We say that $T_X$ is a list of $X\%$ of the elements in the set $T$ ranked by uncertainty (see Eq. 8.1). The idea behind temporal reranking is similar to the idea behind the recency prior: more recent documents are more likely to influence future results. For temporal reranking, $T_X$ is then reranked according to the tweet id.

The intuition behind burst-based reranking is that documents around salient events influence future events and are important to be included in the model. Here, we make use of the bursty nature of social media. We identify if $T_X$ is peaky similar to [94, 188, 288]. We then rank the elements according to them being bursty. We detail the conditions below.

**Peaky** To understand the specifics, we need to define a time series $TS(T)$ of a candidate set $T$ as $(d_1, c_1), \ldots, (d_n, c_n)$. A tuple $(d_x, c_x)$ consists of the date $d_x$ an element in the candidate set was published and the number of elements $c_x$ that were published on date that date $d_x$. We say that the time series $TS(T)$ is peaky, if $\exists (d_x, c_x) \in TS(T), c_x > \mu(\text{TS}(T)) + 4 \cdot \sigma(\text{TS}(T))$, where $\mu(\text{TS}(T))$ and $\sigma(\text{TS}(T))$ are the mean and standard deviation of the counts in the time series, respectively. In natural language, a peaky time series has at least one date were at least four standard deviations more tweets were published than the average.

Figure 8.1 shows an example time series. We can see that one day (2012-02-09) determines that the ranked list underlying this is peaky.
8.2. An Active Learning Approach to Entity Filtering

![Figure 8.1: The number of tweets, published per day for the second time split, i.e., bin of the stream of entities for entity RL2013D01E001.](image)

**Bursty** Similarly, for a tuple \((d_x, c_x)\) in a time series \(TS(T)\), we say a date \(d_x\) is **bursty** if \(c_x > \mu(TS(T)) + 2 \cdot \sigma(TS(T))\). If the \(TS(T)\) is peaky, we rerank \(T_X\) with all elements that were published on a bursty date and then add the other elements. Within the two groups, the elements remain sorted by uncertainty (see Eq. 8.1). In the example Figure 8.1 only one day (2012-02-09) is bursty.

For example, we have a set \(T = \{x_1, \ldots, x_{60}\}\), where

\[
T_{10} = [x_2, x_5, x_{40}, x_{14}, x_9, x_{60}],
\]

and all but \(x_2\) and \(x_{60}\) were published on 13-12-12. The time series

\[
TS(T) = [(12-12-12, 2), (13-12-12, 56), (14-12-12, 2)]
\]

is peaky. The new reranked list is \([x_5, x_{40}, x_{14}, x_9, x_2, x_{60}]\): the uncertainty ordering is kept **within** the groups of elements that were published on bursty dates or not.

**Oracle**

The upper bound for the sampling methods is an oracle sampler. As a true oracle sample has an exponential amount of possible candidate sets to maximize from, we use a greedy oracle. Here, for each iteration, we select a candidate that, when added, maximises the F-score or accuracy on the test set.

8.2.2 Candidate Annotation

In this step of Algorithm 3 (line 17), annotations for the selected candidates are collected. Section 8.3.4 elaborates on how we can simulate the user input.

8.2.3 Model Updating

The training of the model is fast. We therefore decided to **retrain** the model with **every** freshly annotated instance. The instance and its annotation are added to the training set and the model is retrained. As commonly done, the weight for training and new instances is uniform.
8.3 Experimental Setup

In the following we introduce the datasets, settings, and parameters needed to evaluate the effectiveness of active learning for the entity filtering task in a streaming scenario.

8.3.1 Datasets

We use the RepLab 2013 dataset introduced in Section 3.4.2 in Chapter 3 and their annotations for relevancy to an entity.

8.3.2 Document Representation

The tweets are represented as a set-of-words (SoW), using the vocabulary of the training set and the name of the author of the tweet. SoW are with binary occurrence (1 if the word is present in the tweet, 0 if not). The SoW representation was generated by removing punctuation, lowercasing, tokenizing by whitespaces, reducing multiple repetitions of characters (from $n$ to 2) and removing stopwords. This is the same representation as in [229]. The advantage of this approach is that it is not over-engineered and does not make extensive use of additional data or external resources, unlike, e.g., the best performing systems at RepLab 2013 [7]. We used an entity-dependent approach, i.e., we train and test on specific training and test sets for entities.

8.3.3 Streaming Scenario

Evaluating active learning is difficult and costly, since users should provide feedback on each iteration. In a real-life setting, the selected candidate instances would be annotated by users. Those labeled instances are then incorporated into the system and not predicted anymore. Without direct users, the usual approach to model the active learning setting is to take the annotations from the test set. This simulates the user feedback; this is what we do.

In the streaming scenario we model the problem of daily entity filtering for reputation monitoring. For every entity, we sort the tweets in the test set by time and then sort them into bins: every bin containing 10% of the data. For every bin, the model is based on the temporally earlier sampled and annotated tweets.

8.3.4 Settings

For our experiments we use Support Vector Machines, using a linear kernel. The penalty parameter $C$ is automatically adjusted by weights inversely proportional to class frequencies. We use the default values for the rest of the parameters.

---

5 We also considered alternative representations, but these did not outperform this simple set-of-words representation. E.g., the set-of-words representation outperformed a bag-of-word representation that also used linked entities, using 10 fold cross-validation on the training set.

6 We tested different algorithms (Naive Bayes, Decision Trees) and this is the one that obtained the best results in terms of the initial (passive learning) model.
8.3. Experimental Setup

Table 8.1: Runs used in our experiments.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Ref.</th>
<th>Active</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>passive</td>
<td>§8.2.1</td>
<td>no</td>
<td>Passive learning, lower bound</td>
</tr>
<tr>
<td>O</td>
<td>§8.2.1</td>
<td>no</td>
<td>Oracle, upper bound</td>
</tr>
<tr>
<td>best</td>
<td>[214]</td>
<td>no</td>
<td>Best RepLab2013</td>
</tr>
<tr>
<td>RS</td>
<td>§8.2.1</td>
<td>yes</td>
<td>Random sampling</td>
</tr>
<tr>
<td>MS</td>
<td>§8.2.1</td>
<td>yes</td>
<td>Margin sampling</td>
</tr>
<tr>
<td>MS-PRT</td>
<td>§8.2.1</td>
<td>yes</td>
<td>Margin sampling with recency prior</td>
</tr>
<tr>
<td>MS-RRT</td>
<td>§8.2.1</td>
<td>yes</td>
<td>Reranking MS based on recency</td>
</tr>
<tr>
<td>MS-RRB</td>
<td>§8.2.1</td>
<td>yes</td>
<td>Reranking MS based on bursts</td>
</tr>
</tbody>
</table>

For the initial model we have two settings: warm start, where the initial model is based on the training data, and cold start, where we have practically no initial model and the candidate selection for the first bin is always random. For both settings, warm and cold start, we compare two sampling methods, random and margin, for every bin. We sample $N_{test}$ tweets per bin. We report on the effectiveness of single bins, but also on the average effectiveness. Unless otherwise stated, the results are averaged over entities. Since we are dealing with tweets, which is similar to the Tweets2011 dataset (see Chapter 3), we use the same parameter settings for $f_{basic Weibull}$ as in Chapter 7, see Table 7.2. The granularity $g$ is set to one day. We set $X$, the percentage of items to be reranked in a candidate list to 10%.

Table 8.1 provides an overview over the acronyms used for the runs. The passive run is the underlying baseline for active learning; it is based on the training set in the streaming scenario with the warm start and in the batch scenario. In a streaming scenario with cold start it is based on a random sample of $N_{test}$ instances in the initial bin. The best run is the score for the best performing system at RepLab2013. This score is only available for the batch scenario. $RS$ and $MS$ are active learning runs, using random and margin sampling, respectively. $MS-PRT$ denotes margin sampling with temporal priors. Finally, $MS-RRT$ and $MS-RRB$ are margin sampling runs which uses reranking, based on recency, or bursts, respectively. When comparing MS to its extensions, we often call MS “vanilla margin sampling”.

8.3.5 Evaluation

Unless stated otherwise, we use the official evaluation metric from the RepLab2013 Filtering Subtask: accuracy and the harmonic mean of reliability and sensitivity, $(F_1(R, S))$ later discussed in [8]. While accuracy was also part of the official metrics, [229] showed that both metrics are strongly correlated in the active learning setting. Due to the randomness underlying the sampling methods, we report results averaged over 100 runs. Accuracy corresponds to the ratio of correctly classified instances. In scenarios where classes are highly unbalanced (like in the EF task), it is not trivial to understand the effectiveness of a system by measuring accuracy: a system that simply assigns all the instances to the majority class can have a 0.9 accuracy if 90% of the instances correspond to a sin-
gle class in the gold-standard. The alternative metrics used at RepLab2013, reliability & sensitivity (R&S), are more appropriate for measuring how informative a filtering system is. R&S corresponds to the products of precision in both classes and the product of recall scores, respectively. The harmonic mean of R&S tends to zero when the system has a "majority class" behavior, and a high score according to $F_1(R, S)$ ensures a high score for most of the popular evaluation metrics in filtering tasks \cite{8,231}. For the streaming scenario, the oracle is based on the accuracy, because estimating the $F_1(R, S)$ score proved to be too expensive for feasible results.

We use the Student’s t-test to assess the significance of observed differences, using Bonferroni normalisation where appropriate.

8.4 Results and Analysis

In this section we report the performance of the margin and random sampling baseline in Section 8.4.1. We discuss several temporal extensions, be it as a prior to margin sampling in Section 8.4.2 or as reranking in Section 8.4.3. Section 8.4.4 seeks out to identify the best approach.

8.4.1 Margin Sampling

In this section we analyse the effect of margin and random sampling for EF in the streaming scenario and investigate the influence of the strength of the initial model. In particular we ask the question:

**RQ5.1** Does margin sampling improve effectiveness over random sampling, i.e., is it a strong baseline?

We analyse this for strong initial models, i.e., a model trained with a temporally earlier training set, and weak initial models.

Figure 8.2 (Left) shows the development of $F_1(R, S)$ with increasing values $N_{\text{test}}$, for both using a warm start (i.e., using a strong initial model for sampling) and cold start (i.e., building a model based on sampling the first bin). Similar to the batch scenario, margin sampling ($MS$) outperforms random sampling ($RS$) in both settings. For the warm start, we have an exponential increase (with respect to $N_{\text{test}}$) for $MS$ while the increase for $RS$ is merely linear. Margin sampling performs strongly significantly better for $10 \leq N_{\text{test}} \leq 150 (p < 0.005^7)$. For the cold start setting the $F_1(R, S)$-scores are generally low and while $MS$ outperforms $RS$ for most $N_{\text{test}}$, the difference is not significant. Margin sampling benefits from the strong initial model: a strong initial model helps to identify tweets with new topics while the cold start model is still sampling to build up a basic classifier. For both, the cold and warm start, we can see how margin sampling is very close to the oracle after $N_{\text{test}} > 50$, improvements for greater $N_{\text{test}}$ will therefore only be small.

Figure 8.2 (Center) shows the development of $F_1(R, S)$ over time, for the warm and cold start settings, using $N_{\text{test}} = 10 (\approx 5\%)$. For the warm start setting (Figure 8.2b) we see the difference between using $RS$ and $MS$ increasing over time. For higher values of

\footnote{$p = 0.05$ with Bonferroni correction.}
8.4. Results and Analysis

Figure 8.2: Comparison of vanilla margin sampling (MS) with the baselines random sampling (RS), passive learning (passive), and the upper bound (O).  
(Left) Averaged $F_1(R, S)$ vs. $N_{\text{test}}$.  
(Center) Averaged $F_1(R, S)$ over different bins for $N_{\text{test}} = 10$ manually annotated tweets per bin ($\approx 5\%$).  
(Right) Averaged $F_1(R, S)$ over different bins for $N_{\text{test}} = 50$ manually annotated tweets per bin ($\approx 30\%$).

$N_{\text{test}}$ this is more prominent; see Figure 8.2c. MS maintains a stable $F_1(R, S)$-score while the $F_1(R, S)$-score for RS decays over time. For the cold start case, the performance of the passive learner drops a lot: the passive learner is based on an initial model of bin 0 with 10 instances. While the total difference between MS and RS is not significant in the cold start setting, MS performs better on every single bin and, moreover, gets better and better than RS over time. This is even more apparent when annotating 50 samples per bin ($\approx 30\%$); see Figure 8.2f.

Figure 8.3 compares the improvement between using MS and RS in terms of $F_1(R, S)$ (Figure 8.3a), Sensitivity (Figure 8.3c), and Reliability (Figure 8.3b) for individual entities. The reliability of MS is at least as good as of RS for over 95% of the entities (Figure 8.3b); and the harm for reliability is very low. As to sensitivity, MS performs worse than RS for 8 entities but it performs much better for the majority of the entities. If we look at the entities with extreme differences for $F_1(R, S)$ and sensitivity, we have the entity Adele (RL2013D04E145), where the improvement is over 0.6 for both $F_1(R, S)$ and sensitivity. Adele is a common female first name and disambiguation is important. Additionally, the world of pop turns fast and new events, and types of events, are constantly coming in. Terms that were not known based on the training set were GDA (Golden Disk awards), where she confirmed her attendance during the time of the training set. Additionally, a new topic emerged during the time of the test set: remixes with Ellie Goulding. Tweets that were selected for annotation were not only based on those
topics, but also on users with a similar name. Active learning was therefore successful in identifying the rapidly changing topics. In contrast, an entity where random selection worked better in terms of reliability was HSBC (RL2013D02E055). Most candidates for annotation relate to a golf tournament sponsored by HSBC; the margin sampler developed a strong bias towards aspects of golf tournaments (meeting points, winners with new names, etc.). However, this aspect was short-lived and did not contain many tweets: the random sampler did not put a strong bias on this transient topic.

To summarize, margin sampling is a more effective and more stable than random sampling for active learning for EF on tweets. Finally, a strong initial model has a higher effect on the effectiveness of margin sampling than random sampling.

### 8.4.2 Recency Priors

In Chapter 7 we showed that using recency priors for ranking improves the effectivity, in particular on microblogs. In this section, we ask:

**RQ5.2** Does sampling based on recency priors and margin sampling together, outperform vanilla margin sampling with respect to F-score?

As with the previous research question, we again analyse this for strong and weak initial models. Figure 8.4 (Left) shows the development of $F_1(R, S)$ with increasing values $N_{\text{test}}$ for both warm start and cold start. We can see that there is hardly a difference between the two approaches for strong initial models. While not significant, we can see that there is a small difference in the cold start case (Figure 8.4d). We can see how for small $N_{\text{test}}$ margin sampling with temporal recency priors (MS-PRT) performs worse than vanilla margin sampling (MS). For higher values of $N_{\text{test}}$, i.e., stronger models, this changes and MS-PRT performs better.

Figure 8.4 (Center) shows the development of $F_1(R, S)$ over time, for the warm and cold start settings, using $N_{\text{test}} = 10$ (≈ 5%). For the warm start setting MS-PRT performs worse. For the cold start setting (Figure 8.4e) we can see that over time, with an improving model, margin sampling with priors performs better, though not significantly. Figure 8.4 (Right) shows the development of $F_1(R, S)$ over time, for the warm and cold start settings, using $N_{\text{test}} = 50$ (≈ 30%). We can only see a slight tendency that MS-PRT
8.4. Results and Analysis

Figure 8.4: Comparison of margin sampling using a temporal recency prior (MS-PRT) with vanilla margin sampling (MS).

(Left) Averaged $F_1(R, S)$ vs. $N_{\text{test}}$. (Center) Averaged $F_1(R, S)$ over different bins for $N_{\text{test}} = 10$ manually annotated tweets per bin ($\approx 5\%$). (Right) Averaged $F_1(R, S)$ over different bins for $N_{\text{test}} = 50$ manually annotated tweets per bin ($\approx 30\%$).

performs better for both the warm start and the cold start settings. To summarize, on averaged results, we can not see a large difference between the approaches.

Figure 8.5 shows the differences of $F_1(R, S)$ between MS and MS-PRT for all single entities for different, low, $N_{\text{test}}$ in the cold setting. Those graphs paint a different picture, as we would expect to see small and few differences between the entities. However, this is not the case and the positive and negative differences in $F_1(R, S)$ between both approaches are in fact very strong among entities and across all four domains (domain difference is indicated via grayscale). It stands to reason that there are recent entities and non-recent entities. Recent entities are entities where recently published tweets have more impact on the future than for non-recent entities, where the difference to the decision boundary is more important. One example of such a recent entity is PSY (RL2013D04E194), who went viral with Gangnam Style and is very active in social media. Together with the ambiguous name PSY, this is a hard entity but adding a recency prior improves the $F_1(R, S)$ score from $0.0828$ to $0.4905$. An example of a non-recent entity is AC/DC (RL2013D04E159), an old classic, where adding the prior to the margin sampling drops the $F_1(R, S)$ from $0.7$ to $0.0409$. A strong model to distinguish AC/DC from electric terminology is more important than recent changes.

We find that temporal priors can increase effectiveness for specific, recent entities.

153
8. Active Learning for Filtering of Streaming Documents

8.4.3 Temporal Reranking

Microblog data is inherently temporal and both, burst-based and recency methods have been successful for ranking. We apply similar approaches to EF and analyse the answer to:

RQ5.3 Does temporal reranking based on bursts or recency of margin sampled results, outperform margin sampling with respect to $F_1(R, S)$?

As with the previous research questions, we again analyse this for strong and weak initial models. Figure 8.6 (Left) shows the development of $F_1(R, S)$ with increasing values $N_{\text{test}}$ for both a warm start and an cold start. For the warm start, all sampling approaches but RS (random sampling) perform very similar to the oracle and are therefore very close together in general. Burst-based reranking performs better for all ($10 < N_{\text{test}} < 130$) (significantly for $N_{\text{test}} = 50, p < 0.05^8$). We saw earlier that the performance for low $N_{\text{test}} (N_{\text{test}} \leq 50)$ is important. Zooming into $10 \geq N_{\text{test}} \leq 50$, we see that adding burst-based reranking of margin sampled candidate sets (MS-RRB) actually performs better than vanilla margin sampling (MS). Reranking the margin sample based on recency (MS-RRT) underperforms margin sampling significantly ($p < 0.05^9$). Looking at the cold start, temporal reranking (MS-RRT) performs worse than random sampling. However, we can see that while MS-RRB and MS perform similarly, for some $N_{\text{test}}$ vanilla MS performs better, for some MS-RRB. Figure 8.6 (Center) shows the development of $F_1(R, S)$ over time, for the warm and cold start settings, using $N_{\text{test}} = 10$ ($\approx 5\%$). For the warm start setting (Figure 8.6b) we can see that while MS performs better at the beginning, burst-based reranking performs slightly better at the end. Temporal reranking (MS-RRT) performs worse. For the cold start setting (Figure 8.6e) we can again see that while MS performs better at the beginning, MS-RRB performs better for higher $N_{\text{test}}$. Temporal reranking still performs worse than random sampling.

Figure 8.6 (Right) shows the development of $F_1(R, S)$ over time, for the warm and cold start settings, using $N_{\text{test}} = 50$ ($\approx 30\%$). For the warm start setting (Figure 8.6c)

---

\(^8 p = 0.05 \text{ with Bonferroni correction.}^9 p = 0.05 \text{ with Bonferroni correction.}
we can see that MS-RRT and MS-RRB are on the same level. Initially, so is vanilla MS, but with time, both reranking approaches perform better than margin sampling. We can see that they are very close to the oracle. For the cold start setting (Figure 8.6f), while vanilla margin sampling performs better than burst-based reranking, we can see that over time, the burst-based reranking performs better than the oracle (which is in fact an approximation).

To summarize, we saw that with very little training (cold start) and small sample sizes ($N_{\text{test}}$) the burst based reranking does not make much of a difference. Increasing the sample size here, burst-based margin sampling performs better than vanilla margin sampling. For increasing $N_{\text{test}}$, both models converge. We also see that in general, training is important because the margin sampling is too weak. Potentially ignoring the top candidates when reranking based on recency harms. The uncertainty of a sample is much more important than the time of publishing.

However, it does not only depend on the different $N_{\text{test}}$ on when to rerank or not. Figure 8.7 shows the differences of $F(R, S)$ between MS and MS-RRB for all single entities for different, low, $N_{\text{test}}$ in the warm setting. At $N_{\text{test}} = 10$ (Figure 8.7 (Left)) vanilla margin sampling performs better than burst-based reranking for most entities. This makes sense: burst detection on 10 samples is very hard and prone to be unstable. However, we can already see that for one entity, burst detection performs much better. With increasing $N_{\text{test}}$ (Figure 8.7 (Center) and (Right)), we see that in general there is

Figure 8.6: Comparison of reranking approaches (MS-RRT, MS-RRB) with vanilla margin sampling (MS). (Left) Averaged $F_1(R, S)$ vs. $N_{\text{test}}$. (Center) Averaged $F_1(R, S)$ over different bins for $N_{\text{test}} = 10$ manually annotated tweets per bin ($\approx 5\%$). (Right) Averaged $F_1(R, S)$ over different bins for $N_{\text{test}} = 50$ manually annotated tweets per bin ($\approx 30\%$).
not much difference between reranking or not reranking. However, for more and more entities burst-based ranking increases performance (up to 0.4), without harming the performance for other entities (maximally to 0.05). This is not dependent on one specific dimension. Let us now look at two examples: entity The Beatles (RL2013D04E149) and Coldplay (RL2013D04E164). Entity RL2013D04E149 is the entity where burst based reranking performs worst compared to no reranking. Even intuitively, this makes sense: The Beatles are not a current band and therefore not very prone to news and gossip events. Figure 8.8a shows the development of $F_1(R, S)$ over different time bins for $N_{test} = 30$. We can see how the value of $F_1(R, S)$ for burst-based reranking drops to 0 on time bin 5, while it otherwise performs similar to vanilla margin sampling. This is in fact because it is vanilla margin sampling: it does not consider the temporal distributions peaky except for bin 1 and bin 5 and burst-based reranking only reranks if the temporal distribution of the candidate set is peaky. So, why does it go wrong for the peaky temporal distributions? Figure 8.8b shows the temporal distribution of tweets in the test set for entity RL2013D04E149, which is one tweet per date, except for two dates where 2 tweets per date were published and one where 3 tweets were published. The burst-based reranker samples from the three dates which are not actually bursts in our own intuitive understanding. In fact, the tweets from one of the days were by a user declaring his love to his girlfriend by comparing her to The Beatles, in the second burst a user talks about giving The Beatles memorabilia to a friend. Those tweets are not helping: the most useful tweets, also selected by the oracle, are tweets featuring terms like Yoko Ono, Forgery, and parts of songtitles (All my Loving, Yellow Submarine).

Entity RL2013D04E164 is the entity where burst based reranking performs best compared to no reranking. Coldplay is a more recent band, that is still frequently listened to and where all members are still alive. Figure 8.8c shows the development of $F_1(R, S)$ over different time bins for $N_{test} = 30$ limited to entity RL2013D04E164. Here, we can see how burst-based reranking consistently reaches full $F_1(R, S)$, while vanilla margin sampling drops down on bin 1, 5 and 5. Figure 8.8d shows the temporal distribution of tweets in the test set for entity RL2013D04E164. The algorithm identifies the time series as peaky, which is intuitive, and selects tweets from the bursty times, which are all centered around the same time period. In this case, burst-based resampling makes sense.
Content-wise, the tweets were retweets of the type:

\texttt{RT If you like ”Paradise” By Coldplay #RetweetTheSongs}

With the songtitles linked to the bandname, those tweets are useful for the learner, because here again, many tweets about the band feature (parts of) songtitles. Selecting unrelated tweets (as done by unranked margin sampling) misleads the learner, as there are only six unrelated tweets.

### 8.4.4 Is There a Universal Best Approach to Sampling Candidates?

It seems that the sampling method depends on the entity and \( N_{\text{test}} \). To underly this assumption, Figure 8.9 shows which sampling works best for the different entities and for different values of \( N_{\text{test}} \) in the warm as well as the cold start setting. Figure 8.9a shows the selections with initial training. For one, we can see that vanilla margin sampling
Figure 8.9: Selection of approaches with the highest $F_1(R, S)$ for all entities over different $N_{\text{test}}$ for both warm start (8.9a) and cold start (8.9b).

is the predominantly best performing sampling method. We can also see that for low values of $N_{\text{test}}$ vanilla margin sampling is by far not as dominant—with few candidates, every bit of information counts. We can see several entities where burst-based reranking predominantly performs best (like RL2013D02E76) and several where margin sampling with temporal recency priors predominantly performs best (like RL2013D03E87). Figure 8.9a shows the selections without initial training. Here, the vanilla margin sampling is not as frequently selected as the best approach—only for entities in the dimension D04, music. Here, it seems more that the values of $N_{\text{test}}$ determine the best approach: either filtering for an entity with vanilla margin sampling performs best or temporal sampling methods perform best, but then they are mostly based on bursts. As a summary, there are situations where vanilla margin sampling is superior to temporal extensions. In particular, this seems to be entity dependent. However, for several entities, when a little bit of training and knowledge about the entity is present, burst-based sampling performs better, but for higher sampling sizes the performance converges to the performance of margin sampling.

To wrap up the results for the streaming scenario, informed sampling based on margin sampling performs better than random sampling. The particular sampling method is entity-dependent: Informed sampling can be based on vanilla margin sampling, temporal reranking based on bursts, or, for few entities, margin sampling using recency priors.
8.5 Conclusion

The goal of this chapter was the introduction of active learning to entity filtering for streaming documents. Active learning for entity filtering is new, in particular for streaming data. In this chapter we present a streaming scenario for active learning based on the RepLab 2013 entity filtering task. We provide results for the standard baselines as well as for temporal extensions: using temporal recency priors for margin sampling, as well as reranking margin sampled documents based on bursts and recency. We found that for entity filtering, using a strong initial model, active learning with margin sampling improves over passive learning by 30% with only 30 (18%) additional annotations per time period. We find that the best approach for entity filtering is entity-dependent: for some entities vanilla margin sampling works best, for others reranking based on bursts, and for others again the temporal prior works best. This also depends on the learnt model: for very strong models vanilla margin sampling is very good at selecting the right candidates. For very weak models, margin sampling barely manages to find a good candidate and reranking and temporal priors bias this in the wrong direction. For intermediate models, temporal margin sampling can perform better than vanilla margin sampling.

Future work should investigate automatic detection of the right sampling methodology for each entity, similar to query classification [125]. This can again be semi-automatic. Larger datasets per entity over a longer period of time can give insights into the performance of each method and how the best candidate selection approach might vary over time. This experimental setup is a simulation of active learning. Online active learning with appropriate user interfaces should bring new insights. Additionally, user interfaces can feedback expert knowledge in a different, non-binary way, be it in form of term clouds or potential events.