arXiv:1204.3362v1 [cs.IR] 16 Apr 2012

Event based classification of Web 2.0 text streams

Andreas Bauer
Media Computer Science
University of Regensburg
andreas.bauer@extern.uni-regensburg.de

Christian Wolff
Media Computer Science
University of Regensburg
christian.wolff@sprachlit.uni-regensburg.de

Abstract

Web 2.0 applications like Twitter or Facebook create a continuous stream of information. This demands new ways of analysis in order to offer insight into this stream right at the moment of the creation of the information, because lots of this data is only relevant within a short period of time. To address this problem real time search engines have recently received increased attention. They take into account the continuous flow of information differently than traditional web search by incorporating temporal and social features, that describe the context of the information during its creation. Standard approaches where data first get stored and then is processed from a persistent storage suffer from latency. We want to address the fluent and rapid nature of text stream by providing an event based approach that analyses directly the stream of information. In a first step we want to define the difference between real time search and traditional search to clarify the demands in modern text filtering. In a second step we want to show how event based features can be used to support the tasks of real time search engines. Using the example of Twitter we present in this paper a way how to combine an event based approach with text mining and information filtering concepts in order to classify incoming information based on stream features. We calculate stream dependant features and feed them into a neural network in order to classify the text streams. We show the separative capabilities of event based features a base for a real time search engine.

Keywords: information retrieval, text mining, event processing, web2.0, text streams, real time search, neural network, stream features

1 Introduction

Instantaneous information sharing offered by services like Twitter, Facebook or Tumblr led to a major increase of user generated content. This creates a continuous stream of information which poses new challenges regarding processing, analysis, information retrieval and filtering. While traditional search engines like Google, Bing or Yahoo are focused on delivering information which covers the whole range of relevants facts regarding a given search query, real time search engines like new Google+, Twitter Search or topsy.com intend to deliver insight into the continuous information stream right at the moment when the information is created, i.e. the tolerance for latency compared to traditional search engines is very low[15].

In this paper real time search engines are considered as being a system that removes irrelevant items from a continuous stream of data. Hence real time search engines are very similar to information filtering systems as described in [2]. We do not consider the type of real time engines, where real time means the instantaneous delivery of items from a corpus, e. g. Solr.

In this paper we want to layout the basis for an event based information filtering system that analyzes real time text streams like the ones produces by Twitter or Facebook. Hence our approach can be considered as the groundwork for a real time search engine. The contributions are made with this paper

- Introduction of atomic information events
- Mapping of text streams to different information event types

- Calculation of stream based event features
- Training and evaluation of neural networks with stream based features
- Evaluation of the performance of the trained networks

The paper is structured as follows. In chapter 2 we give an overview of the already existing literature and research. Chapter 3 defines the problem definition, while chapter 4 provides the theoretical basis. In chapter 5 we show the application and evaluate our approach.

2   Related work

This paper combines several research areas. [2] provides the basic framework for our research. The paper defines the basics of information filtering and introduces basic concepts we also use in this paper, e.g. analysis of text streams, standing queries, removal of information from the stream, user profiles, etc. [20] introduce a way of real-time, top-k and profile based information filtering with sliding time windows based on the traditional tf/idf weighting method. The goal and approach of their paper is similar to this paper's goal. Our approach differs as we use neural networks to learn measures of interest and use stream based features for calculating similarity measures. "Topic Detection and Tracking" addresses similar questions like information filtering. Noteworthy in the context of Twitter topic detection is [8]. They use an aging theory based approach for judging new Tweets. [10] also use a similar setup for detecting interesting Tweets. But first they use incremental Naive-Bayes-Classifier and second they do not focus on the usage of events and stream features.

Stream processing and data stream mining is also closely related to the topics of this paper. [3] describe a data streaming approach for sentiment analysis in Twitter data. They apply multinomial naïve Bayes, stochastic gradient descent and Hoeffding trees in order to identify sentiments from Tweets. In contrast to this paper we focus on filtering Tweets, we use neural networks and in our setup the trained networks are only applied to a short sliding time window and then are rebuilt from scratch. [17] combines incremental decision trees and streamed text features in order to filter Tweets from Twitter lists. It also includes real time aspects as it uses the Twitter Streaming API, but its goal mainly to filter Tweets from group lists with less frequent changes.

Real time search and real time information filtering is addressed in [32], who describe a system where they use current information available about the Twitter stream and the user, in order to build an information retrieval system based on tf/idf, which filters tweets from an incrementally growing Tweet corpus. Also [27] introduces a way using trending detection within the Twitter stream in order to filter unrelevant information. The Microblog track from the TREC 2011 conference is also related, but this track focused on the retrieval of Tweets from a non-dynamic corpus. The task was also called real time retrieval, but more in the sense of finding all relevant Tweets up to a given timestamp. This contrast from our approach as we focus on real time filtering. Also related to this paper is [14] who propose a learning to rank approach for Tweets. This is similar in terms of feature engineering and machine learning. But we use a rather dynamic, continuous way of learning and we use an artifical target function, while they propose the analysis of a static Twitter corpus with manual annotated gold standard.

3   Problem definition

The continuous flow of information from Web 2.0 sources opens the application for a new type of search engines that deliver results based on content that gets generated during the information seeking episode and are not based only on a periodically updated document corpus like in classic search engines. The so called real time search engines deal with a continuously changing document corpus and address the problem that they permanently have to rank and classify incoming new items and present the updated result immediately to the user. This is in contrast to classic web search where the ranking of a document is precomputed in some way based on a given relevance schema and a static corpus size at the time of computation. Of course current web search engines update their corpus many times each second and constantly recalculate the metrics for their search engine, but this applies to their overall corpus, i.e. the overall corpus is growing each second but it's not assured that the corpus regarding a real time search need is also growing every second. Regarding a real time search episode the content crawled by classic search engines doesn't update so frequently thus only the aforementioned Web 2.0 stream can provide new documents for the corpus of the real time search. Real time search engines incorporate a *micro corpus* which is being update during and is only relevant for the time of the

user search. The fact that classic web search engines blend real time search results into their classic web search results shows the importance of real time content. 1 shows the decay in relevance of information events over time. Hence a processing close to the creation of an information event is sensible.
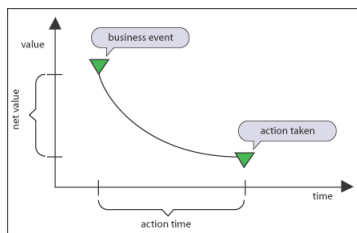

Figure 1: value time curve [18]

Our research focus in on the continuous removal of irrelevant data from a text stream. The information filtering episode we consider is that a user is interested in a certain topic. It is not given that the user has an elaborated personal web profile or if she has one then it might not be related to the information filtering episode. E.g. the user mainly tweets about machine learning and data mining, but for her information filtering episode she is interested in the US preelections. Thus her personal Twitter profile won't necessarily match.

While the interaction in the Web 2.0 entails not only text, but also multimedia, we focus only on textual information streams in this paper. The properties of a continuous text stream is the main challenge for real-time search engines and demands special attendance. We focus on the Twitter stream due to its availability, but in later research we want to apply our approach also to other text streams.

Characteristics of Web 2.0 text streams based on results from [12] and [39] show the peculiarities, which have to be considered.

- Recency
- Timeliness
- Data Volume
- Temporal Validity
- Temporal relevance
- Social interaction
- Shortness
- Dynamic corpora (terms, documents)

To our knowledge only few information is publicly

Aspect	Real time search	classic search
Corpus	dynamic	static
Corpus update	continuous	periodic
temporal relevance	short	arbitrary
Query modification	rare	frequent
Content generation	spontaneous, ad-hoc	elaborated
Document length	short	arbitrary

Table 1: Comparison features real time vs. classic web search [39]

available on how real time search engines assess incoming information. Google's realtime search - before it was closed down in July 2011 and later incoporated into Google+ - was supposed to use an algorithm similar to PageRank, where not the link structure was taken into account, but the reputation of the user [38]. http://www.topsy.com also relies on a influence factor of the content creating users. Other search engines like SocialMention.com, kurrently.com or 48ers seem to take into account key word filtering as well as social features, but nothing is known about their approaches. The measure of all real time search is Twitter itself. Their corpus is of course sublime, as they have all the Twitter data, but we think our approach adds an interesting aspect to this topic.

As a summary table 1 summarizes the main differences between real time and classic web search.

In order to add some new ideas to this interesting and rising topic we like to introduce an intuitive approach for ranking real time content, which combines document-features as well as term weighting with text mining and event processing. While the first three aspects are well established in the areas of information retrieval and ranking, the latter allows to react on real time content and to determine features for incoming items as they occur. The combination of these four research areas allow us to address the problems that are posed by the characteristics of Web 2.0 text streams 3.

We want to introduce an event based approach to tackle these problems. This approach uses methodologies from event processing which apply naturally to the demands of streamed text data. The advantage of event processing is that you do not have to build tem-

poral data structures and window based calculations yourself, but you can use sophisticated stream engines which offer advanced functionalities like sliding windows or pattern matching. We combine this with the foundations of information retrieval and information filtering to answer questions 1-4.

There are many research papers on analysing ([5], [23]), classifying ([17],[3]) or ranking (TunkRank, etc.) with continuous text streams - foremost twitter - , but all have in common that a several human estimators had to provide classes or rankings in order to train the models. We try to overcome this task by using stream properties for training the ranking function and for ranking the text stream snippets. This is foremost important as our research focus lies on ad-hoc, short-term filtering and monitoring tasks for Web 2.0 text streams where we assume that during the time of the information episode no elaborated user preference model is available. This is in contrast to user profile based search or collaborative search, where you can use historical data of the user itself or a group of users in order to build a user model that can be used in the search. We exclude this fact in this paper and save the analysis for later research. In this paper we assume that it is not always possible to include a user profile because either the profile is simply not available, not yet elaborated enough to be taken into account or is not related to the filtering task. Hence the proposed approach shows one component for a real time information filtering systems. The query matching part also will be presented in another research paper.

4 Event based information filtering for continuous web 2.0 text streams

In the next section we want to introduce the basics for an event based information filtering and retrieval system for real time text streams from web 2.0 sources, show why an event driven approach is well suited for this.

In the first part of this chapter we show how to map a raw tweet onto several independent event types, which in turn are fed into individual streams for the analysis. In the second step we show how you can apply quantitative and qualitative stream filtering and pattern matching methods in order to extract relevant events from the several streams. Then in the third part we show how we use these filtered events for ranking single events within the stream and how we use this approach for classifying tweets into several categories. The evaluation and examination follows in section 6.

Event based information filtering Before we start to explain our event based system, we want to clarify the notion of the term *event* we use through out this paper. While *event* is a very generic term and is overloaded with different meanings in different research areas, we use the notion coming from the research area of *event processsing systems*. This foremost based on the definition of [31] and [26] and is already widely used in areas of logistics or financial services. We want to introduce the concept of atomic information events for information retrieval and filtering.

In our approach we consider the text stream information as consisting of smaller, simpler events, i.e. each Tweet, Blog post or Facebook update is made up of several tinier events that belong to different event classes like token events, location events or link events. This allows us to analyse not only the stream of incoming "raw events" like Tweets, but also we can analyse the underlying "simple" events that are the building blocks of the raw event and combine them to more complex events.

One might argue that using the term *event* instead of the term *token* does not have any relevance for text mining and information filtering/retrieval. But we emphazise that the use of the event metaphor offers several advantages. First the term event is a semantic construct, which underpins the temporal and dynamic character of data stream. It suitable to describe the state of informations from their time being created to the time it makes its way to *permament* part of the web. During this time the information event's signal is the strongest and offers the most potential for analysis. Second for a real time filtering system it is sensible to only consider a limited time horizon as relevant. Hence events that leave this horizon have lost the majority of their importance. This is supported by the temporal semantics implied with events. Third events are only evaluated once within an event processing network, i.e. after they have entered the network they run through all standing queries and patterns that are in place and then leave the processing workflow. This supports instantaneuos processing of the data stream. Fourth an event is not bound to a distinct processing step. The system can be flexibly extended as events are not bound to a specific event processing agent. Last the event based approach offers the ability of processing information right in its natural and actual context. The context for information, that has already been persisted, has to be artifically recreated, i.e. there is a notable amount

of overhead for putting the information in its original state. Thus the usage of information events is highly efficient compared to persisted information items and allows the creation of executable knowledge right at the creation time of information.

4.1 Event mapping and stream creation

First we build several independent streams from the raw torrent of tweets. For this we process the structured metadata part of the tweet and the un/semistructured part of the tweet separately. The metadata information such as location, followers or friend count, number of tweets or local time can be mapped directly to distinct event types. These events are all composed of the following parts [31]:

- header / description attributes:
    unique event identifier
    timestamp and time granularity
    event source
- payload: actual event information

For our event based information filtering/text analysis system we map a Tweet onto the following event types

- Token score event
- Link events
- Retweets events
- Hashtags events
- Cooccurrence events
- Metadata events: status, follower count, followee count

The mapping process of a tweet is split into two parts and is performed by Event Processing Agents (EPA) [31], which are placed in the text stream. The one for the structured, directly accessible metadata part, i.e. information that can be directly extracted from the tweet without further processing. In theory you could spare this step and operate directly on the properties of the raw event, i.e. the Tweet. But in order to have clear semantics which support and ease the definition of the processing logic for the Twitter stream it is recommended to map the information on separate event types. Additionally this offers the advantage that you can directly operate on the event information without additional filtering for the desired attribute and second you avoid unnecessary payload overhead, which can have impacts on performance when you take into account that several thousand events per second are processed.

The second step deals with extracting and mapping of the un/semistructured text part of a tweet. In this case un- or semistructured means, that the information within the text itself has to be made processable for an event processing engines. In order to access the information buried within the text, we apply on the one hand standard text mining preprocessing and on the other hand we extract additional semantics from the text based on the characteristics of tweets. The first step is used for creating linguistic base event types like TokenEvents or CooccurrenceEvents and includes tokenization, stemming and stop word removal. At the end of this process we get a normalized token which you can use to populate, e.g. a TokenEvent, with its payload.

The second step uses known semantics of Tweets that were introduced by Twitter or Twitter users to provide additional information. This includes mentions (@ sign), hashtags (#), retweets and links. We use regular expressions in order to extract the information. Each encountered item is mapped onto the corresponding event type. The aforementioned semantics are typical for Twitter, but this approach can be used for every event source that provides extractable semantic features.

At the end of the process we end up with several independent information streams which can be analyzed separately, but can also be joined by their common event attributes, e.g. the unique of the Tweet the events were derived from, or finally employed for event pattern matching.

4.2 Leveraging stream properties and event patterns for ranking and classification

In this section we want to describe how to use stream features for real time classification purposes.

In our analysis we employ event processing techniques to select and build features for the machine learning algorithm. The features are then fed into an artifical neural network (ANN). [16] gives an overview of the advantages of using Neural Networks for text classification. We use a neural network because they are well suited for pattern detection, are capable to deal with dynamic and incomplete data. All three properties are typical for our setup, as we look for patterns in a sampled and dynamic text stream.

After the network is trained the new events, that have arrived within the defined time window, are assessed and the top percentile is kept for further evaluation. The results of this step can be considered again as new events that can be fed into other event processing agents. Combined with events from EPAs, that deal with query evaluation or context definition, the information events can be combined to a final result that is presented to the user. But to keep the focus clear for this paper, we concentrate on the description of the basics of the event based systems and one of its application - the usage for training machine learning algorithms.

4.2.1 Defining quantitative measures for text stream analytics

By mapping the features of a Tweet onto separate event types and feeding the events into separate streams, we now can directly operate on the events with established stream processing methods. These entail the calculation of stream statistics on sliding windows, detecting patterns like drops and burst, recognition of event sequences and correlating of different streams. Furthermore we can use the capabilities of stream engines to provide statistical information (standard deviation, variance) on different properties of a stream. Combining these features we are able to create a focused research corridor which allows us to judge Twitter messages as they occur. In this paper we mainly focus on volume based features.

Stream characteristics An important factor for any filtering system is its real world applicability. In our experimental setup we used an event frequency of 100 Tweets per seconds. The system is implemented in Java and uses a maximum heap size of 4GB. The experiments were conducted on a Mac with a Dual 3.2 ghz processor. Throughout the experiments the event speed could be kept constant. Besides the training time of the neural network, the delay for processing the incoming events, i.e. the time the event processing agent needed to map the stream features onto training features, was about 2 seconds.

Twitter in reality processes approx. 4000 Events per seconds. At peek times it is about twice as much. In general approx. 40% of all Tweets are English. Hence a system has to process 1600 to 3200 Tweets per seconds. As we only had access to the Twitter gardenhose with 10% of all Tweets and as only 40%

of those Tweets are English, we chose to use an event speed of 100 Tweets per second. For testing purposes the system was also able to go up to 500 Tweets per second in its basic setup, i.e. the multithreading capabilities of the stream engine were not fully exploited yet.

Features In order to learn a model which can be used for classification, we have to select and calculate appropriate features. The possible features that can be used are already limited by the characteristics of a single Tweet. Thus our features do not fundamentally differ from those, e.g. used by [30].

The features are not calculated on a static document corpus but are based on sliding time windows, i.e. all features reflect their state within the defined sliding time window, e.g. if we define a sliding time window length of 30 seconds the features represent their state within those 30 seconds. This additionally covers problems with concept and topic drift within a stream as we steadily adapt to the new stream characteristics.

We have *event specific features*, like hashtag/link presence and bursty features count, which are directly dependent on the text of the event, as well as *stream specific features* like token score. The latter are not directly calculated based on the event itself, but depend on the stream state for the entierty of a given event type, i.e. the token score of a single term within a Tweet is derived from the stream state of that token within the sliding time window. In a textual analysis like this the score of a token depends on the occurrences of this token within the considered sliding time window.

- Score for 5 token within a Tweet
- Scale normalized score for follower, friend and status count
- Presence of link
- Presence of hashtag
- Frequency variation for 5 token (compared to window two minutes ago)

4.3 Neural network setup and topology

The next step in our approach is to feed the features into an artifical neural network (ANN) in order to learn the target function for the current inspected time slot.

For this we first have to define a target function which the neural network can learn. As it is usually impossible to provide an adequate amount of labelled data for classification in the desired time frame of only a few seconds, we chose an artifical target value, which can be defined for every single tTweet. We also prefer this approach, as in reality you cannot rely on extensive user labelled data (unless you are Google). In this setup we use the presence of a Retweet of a Tweet within a sliding time window of 120 seconds, i.e. the Tweet itself and its Retweet occur within the aforementioned time window. The interpretation of a Retweet being a measure of interestingness is also supported by [30], [37] and [29].

After we have trained an classifier for the analyzed time window, we apply the classifier to new Tweets. In addition the new Tweets are used to train a new classifier from scratch. With this procedure we can provide a continuous classificagion of incoming Tweets.

The size of the input layer of the neural network is defined by the number of features described in the section above. We use a 3-layer neural network where the input layer consists of 15 input features, a hidden layer with 10 neurons and an output layer with a single neuron. The size of the hidden layer was chosen by taking the average of the output of an incremental pruning approach, i.e. for identifying a reasonable size of the hidden layer several several runs were conducted which included a pruning step. The learning algorithm is resilient backpropagation, the activation function is sigmoidal for the hidden layer and SoftMax for the output layer. The latter is due to the fact that we want to treat the outcome as the posterior probability of the Tweet belonging to a class. The error function is linear as this fits well to a softmax activation function and the penality calculation is based on cross entropy [4]. We used the Encog Neural Network library by Jeff Heaton.

Overview of the employed features.

- Hashtag indiciator (binary)
- Link indicator (binary)
- Score of top 4 tokens within a Tweet (numeric)
- Frequency variation of sliding time window for top 4 tokens
- Scale normalized amount of Tweets by user
- Scale normalized amount of friends by user
- Scale normalized amount of followers by user

Measure	Random	Real world data
f-measure avg.	0.4556	0.6515
precision avg.	0.4991	0.6085
recall avg.	0.4396	0.7024
Number of observed sliding windows	20-200	20-200
Training samples avg.	8147	9917

Table 2: Results of experiments

- Scale normalized length of Tweet

The output layer has the size of one, as we deal with an information filtering problem where it is the main goal to devide the value of a piece of information into categories of relevance and non-relevance for further processing. So with the target function defined above we labelled data as relevant or not relevant and use this information for training.

The training and ranking for sliding time windows will work as follows:

1. Define a sliding window size $length(t_i)$
2. Calculate streams features for sliding window t_i
3. Normalize each features by calculating the scale normalized value
4. Define a target value for each Tweet (e.g. retweet as as sign of significance and interestigness)
5. Setup a new ANN for sliding window t_i
6. Split the captured events of sliding window t_i into test and training test
7. Train, crossvalidate and test the ANN
8. Calculate a score for each captured event of sliding window t_i using the trained ANN
9. Return a sorted list of the scored events
10. Combine the scored events with query in order to filter the relevant tweets

5 Experiments and Evaluation

In order to show the applicability of our approach we evaluate the method described in the section above on selfsampled Twitter data.

We show the f-measures, precision and recall curves for our setup over an information filtering episode.
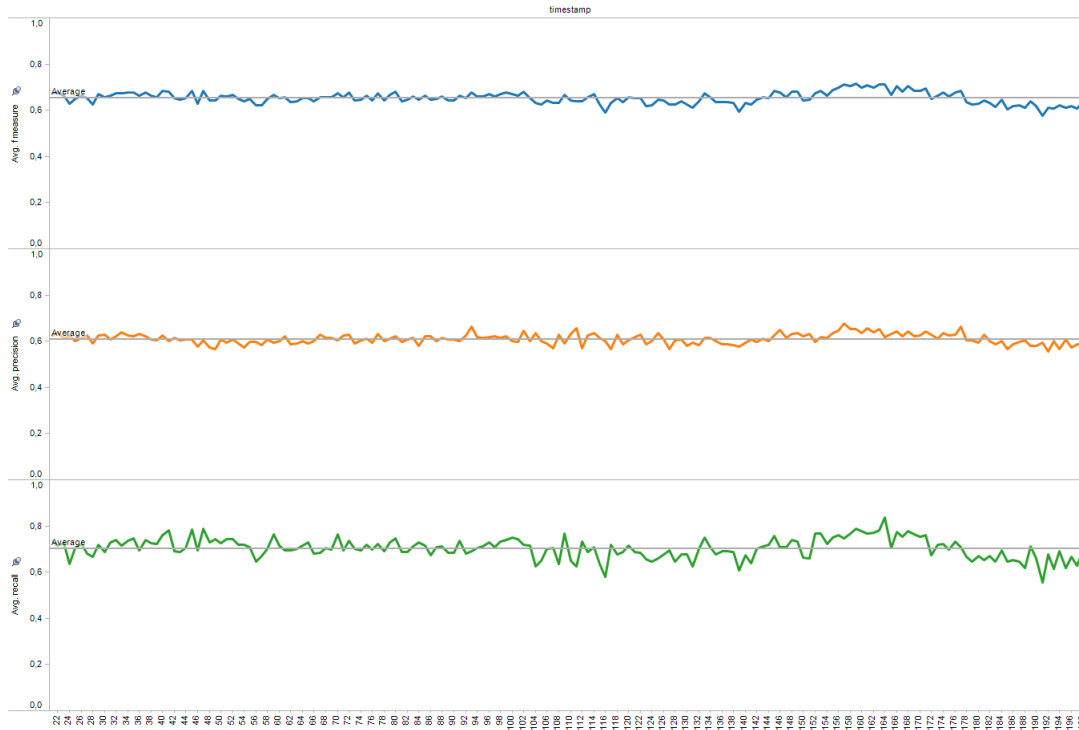
Figure 2: Measure for experiments with real data

For this paper we chose a corpus of Tweets collected on March 2nd 2012 8.00 - 18.00 CET. The corpus contains 839.095 Tweets that we collected with keyword tracking from the gardenhose streaming API of Twitter. For repeatability purposes we saved the Tweets to a database. Furthermore we ran a language detection algorithm [36] on it, as we only want to use English Tweets.

The whole setup works in a streamed fashion due to its natural event based character. We defined a sliding time window of 120 seconds. To stabilize the stream the output of the sliding window will be fed into the neural network every 10 seconds. The neural network gets trained from scratch for every sliding time window again. As the ratio of positives Tweets (Tweets that were retweeted within the sliding time window) and negative Tweets is very skewed (approx. 2%:98%), we use oversampling in order to overcome the imbalance in the dataset. [41] showed that this approachs works despite its potential drawbacks like overfitting. To avoid overfitting we split the samples coming from the sliding time window into three independant sets, one for training, one for testing and one validation. We used a 5-fold cross validiation approach and one fifth of the data was used for the test-

ing of the trained model. Additionally we apply early stopping of the neural network after two increase in a row of the training error. And finally we set a maximum training time of 10 seconds. All measures are intended to keep a good performance for our setup and within the real time constraints of the setting. A summary of the result can be found in table 2

Finally we use the trained network to evaluate newly arriving Tweets. The outcome of the neural network for each new Tweets can be considered as the probability of the new Tweet being similar to the pattern of the training set. We do not consider the outcome as the probability of a Tweet being retweeted, but as the similarity of a Tweet to another Tweet that has been retweeted and thus has been considered interesting. This result can be the input for further processing with the event processing network and hence can be combined with results from other event processing agents.

To have a reasonable comparison of the performance of the system, we show the f-measures, precision and recall of our setup compared to the measures of naive random approach, where the target values are the same like in the runs with real data, but the features
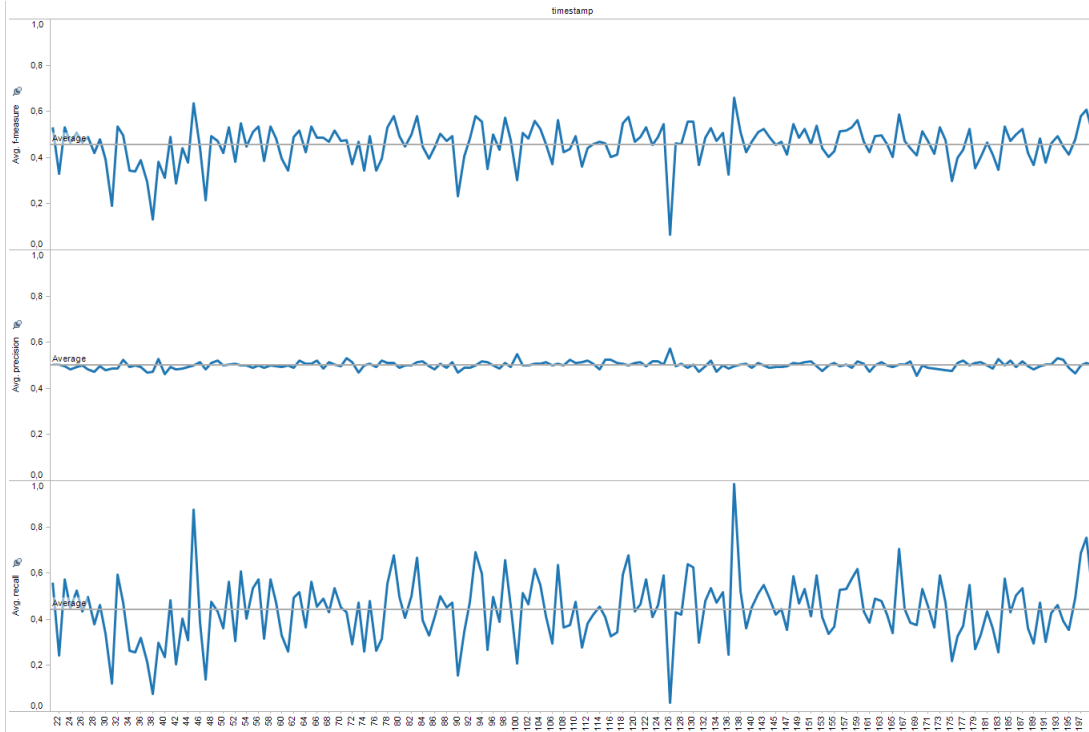
8

Figure 3: Measures for random experiment

were initialized randomly on scale 0 to 1. 2 shows that our approach performs better than the random setup 3. Even though we have to mention that the training and test errors between our system and the randomized approach not as distinct as the f-measure, it can be proved that the selected features are appropriate for the selected task. Furthermore it becomes obvious that our approach offers a stable output performance over the evaluated time frame (we only depict the results of sliding window number 20-200, in order to keep the graphs comprehensible). In 3 you can see that precision values are almost exactly 0.5 what you can expect by guessing a binary classification. The recall values of the random approach differ extremly compared which underpins the assumption that there is a regular pattern within the stream that can be learned. This can be proved by the higher and more regular values produced from our approach.

The last indicator that our approach works is the Cohen kappa statistics. Over the recorded sliding time windows the kappa value for our approach was approx. 0.4, which indicates according to [24] a fair to moderate classification performance. To verify the setup the kappa for the randomized data had an expected kappa statistics average of 0.

6   Conclusions and Future Work

In this paper we presented a event based approach for training an artifical neural network with information events. We showed how to map a Tweet on different event types, analyse these information events with event processing techniques and how to combine this with machine learning approaches. We showed how to calculate the features for the neural network and evaluated the described setup using standard performance criteria.

It could be shown that despite being a very noisy media the information from the Twitter stream can be used to train a neural network model which gives reasonable f-measures. Even though the kappa statistics values are only fair to moderate, our approach gives good inidicators for being usuable in an event based filtering system. In the future we want to elaborate the distinctiveness of different feature set, apply other learning algorithms like SVM or online learning algorithms. Besides this *computational approach* we want to investigate the heuristic based pattern approach for the analysis of information events, i.e. we want to build a net of standing queries that look for interesting events within an information stream.

In this scenario we selected a measure of potential interestigness, but the target function is not limited to this scenario. The experiments showed the separative power of our event based features. Thus we will try different target functions, e.g. to detect spam. Finally as this method gives only a relevance estimate for all Tweets within the stream, we want to incorporate further filtering methods in order to adapt the system to the user and deliver only information events that are relevant for a particular user. We also intend to enlarge this approach to more elaborate web content like blogs or news feeds

References

[1] O Alonso, J. Strötgen, R. Baeza-Yates, and M Gertz. Temporal information retrieval: Challenges and opportunities. *TWAW Workshop, WWW*, pages 1–8, 2011.

[2] NJ Belkin and WB Croft. Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

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

[4] Christopher M Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1 edition, January 1996.

[5] D Boyd, S Golder, and G Lotan. Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1–10, 2010.

[6] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, April 1998.

[7] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[8] M Cataldi, L Di Caro, and C Schifanella. Emerging topic detection on Twitter based on temporal and social terms evaluation. *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, pages 1–10, 2010.

[9] S Chakravarthy and R Adaikkalavan. Events and streams: harnessing and unleashing their synergy! *Proceedings of the second international conference on Distributed event-based systems*, pages 1–12, 2008.

[10] Arifah Che Alhadi, Thomas Gottron, Jerome Kunegis, and Nasir Naveed. Livetweet: Monitoring and predicting interesting microblog posts. In *Proc. European Conf. on Information Retrieval Demonstrations*, 2012.

[11] J Chen, R Nairn, L Nelson, M Bernstein, and E Chi. Short and tweet: experiments on recommending content from information streams. *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1185–1194, 2010.

[12] A Dong, R Zhang, P Kolari, J Bai, F Diaz, Y Chang, Z Zheng, and H Zha. Time is of the essence: improving recency ranking using Twitter data. *Proceedings of the 19th international conference on World wide web*, pages 331–340, 2010.

[13] A Dong, R Zhang, P Kolari, J Bai, F Diaz, Y Chang, Z Zheng, and H Zha. Time is of the essence: improving recency ranking using Twitter data. *Proceedings of the 19th international conference on World wide web*, pages 331–340, 2010.

[14] Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and Heung-Yeung Shum. An empirical study on learning to rank of tweets. In *COLING '10: Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, August 2010.

[15] Miles Efron. Information search and retrieval in microblogs. *Journal of the American Society for Information Science and Technology*, 62(6), June 2011.

[16] M. Govindarajan and RM Chandrasekaran. Classifier Based Text Mining for Neural Network. *Proceedings of World Academy of Science, Engineering and Technology*, 2007.

[17] B Güç. Information filtering on micro-blogging services. *e-collection.ethbib.ethz.ch*, January 2010. URL http://e-collection.ethbib.ethz.ch/eserv/eth:1802/etl

[18] Richard Hackathorn. Real-time to real value, January 2004. URL http://www.information-management.com/issues/20040101/7913-1.html. Accessed 02.03.2012.

[19] P Haghani, S Michel, and K Aberer. Evaluating top-k queries over incomplete data streams. *Proceeding of the 18th ACM conference on Information and knowledge management*, pages 877–886, 2009.

[20] P Haghani, S Michel, and K Aberer. The gist of everything new: personalized top-k processing over web 2.0 streams. *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 489–498, 2010.

[21] Bernard J Jansen, Zhe Liu, Courtney Weaver, Gerry Campbell, and Matthew Gregg. Real time search on the web: Queries, topics, and economic value. *Information Processing & Management*, pages 1–16, February 2011.

[22] BJ Jansen, G Campbell, and M Gregg. Real time search user behavior. *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, pages 3961–3966, 2010.

[23] A Java, X Song, T Finin, and B Tseng. Why we twitter: understanding microblogging usage and communities. *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65, 2007.

[24] J.R. Landis and G.G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174, 1977.

[25] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, Vol. 3: No 3 3 (3):225–331, 2009.

[26] David Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Professional, 1 edition, May 2002.

[27] Michael Mathioudakis and Nick Koudas. TwitterMonitor: trend detection over the twitter stream. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*. ACM Request Permissions, June 2010.

[28] K Mouratidis and H.H. Pang. An incremental threshold method for continuous text search. *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 1187–1190, 2009.

[29] N. Naveed, T. Gottron, J. Kunegis, and A.C. Alhadi. Bad News Travel Fast: A Content-based Analysis of Interestingness on Twitter. *websci11.org*, 2011.

[30] N. Naveed, T. Gottron, J. Kunegis, and A.C. Alhadi. Searching microblogs: coping with sparsity and document quality. *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 183–188, 2011.

[31] Peter Niblett Opher Etzion. Event Processing in Action. pages 1–386, July 2010.

[32] O Phelan, K McCarthy, and B Smyth. Using twitter to recommend real-time topical news. *Proceedings of the third ACM conference on Recommender systems*, pages 385–388, 2009.

[33] C.J.C.B.R.R. Quoc and V. Le. Learning to rank with nonsmooth cost functions. *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, 19:193, 2007.

[34] D Ramage, S Dumais, and D Liebling. Characterizing microblogs with topic models. *International AAAI Conference on Weblogs and Social Media*, 2010.

[35] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979. ISBN 0408709294.

[36] Nakatani Shuyo. Language detection library for java, 2010. URL http://code.google.com/p/language-detection/.

[37] B Suh, L Hong, P Pirolli, and E.H. Chi. Want to be retweeted? Large scale analytics on factors impacting retweet in Twitter network. *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 177–184, 2010.

[38] David Talbot. How google ranks tweets. Web, January 2010. URL http://www.technologyreview.com/web/24353/page1/.

[39] Jaime Teevan, Daniel Ramage, and Merredith R. Morris. #TwitterSearch: a comparison of microblog search and web search. In *Proceedings*

of the fourth ACM international conference on Web search and data mining, WSDM '11, pages 35–44, New York, NY, USA, 2011. ACM.

[40] C. Tryfonopoulos, M Koubarakis, and Y. Drougas. Information filtering and query indexing for an information retrieval model. *ACM Transactions on Information Systems (TOIS)*, 27(2):10, 2009.

[41] G.M. Weiss, K McCarthy, and B. Zabar. Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs. *Proceedings of the 2007 International Conference on Data Mining*, pages 35–41, 2007.

[42] Q Zhao and P Mitra. Event detection and visualization for social text streams. *proceedings of ICWSM'2007*, pages 26–28.