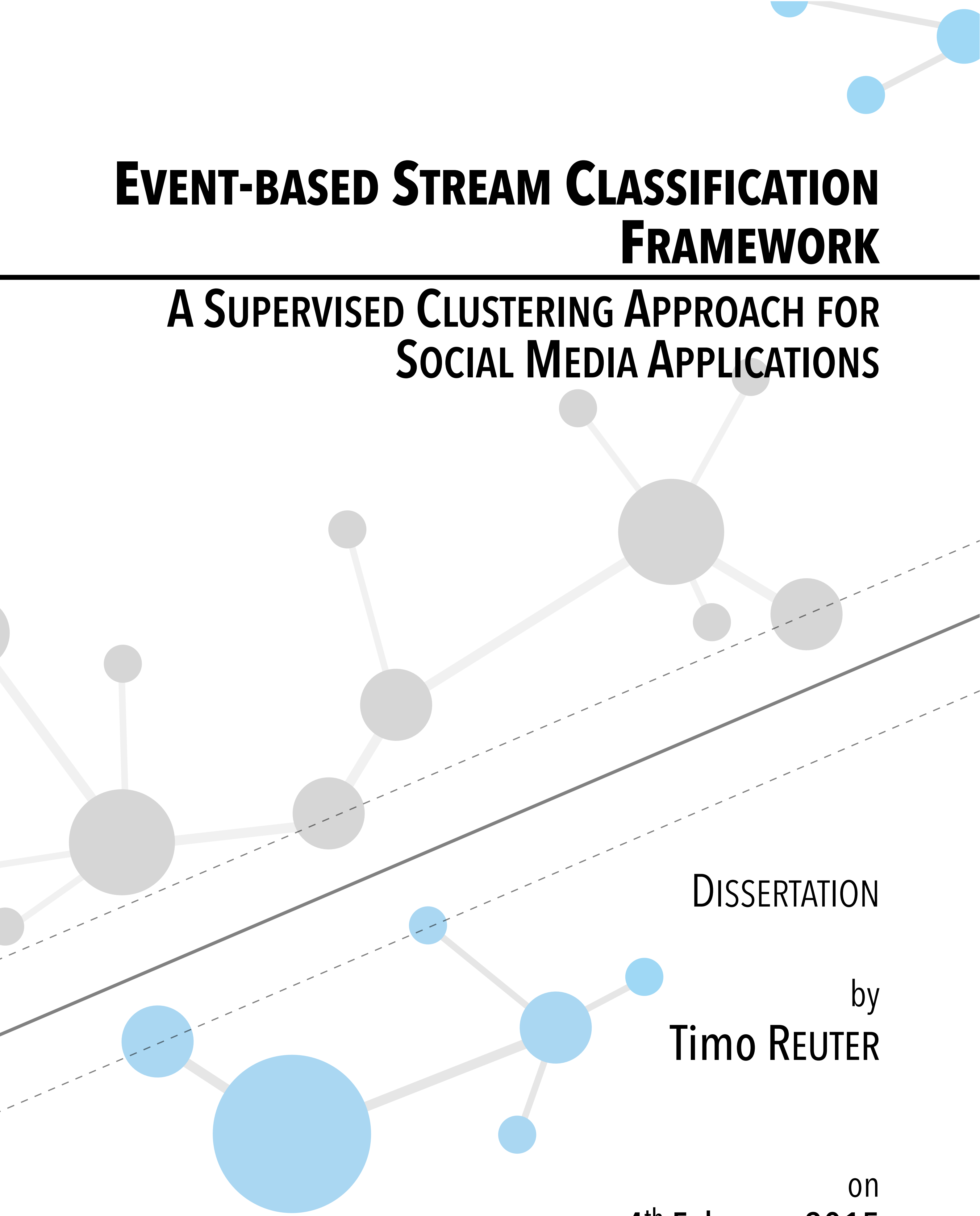
Event-based Stream Classification Framework

A Supervised Clustering Approach for Social Media Applications



Dissertation

by

Timo Reuter

on

4th February 2015

Gedruckt auf alterungsbeständigem Papier nach ISO 9706

Universität Bielefeld

Technische Fakultät

Event-based Stream Classification Framework

A Supervised Clustering Approach for Social Media Applications

Dissertation

zur Erlangung des Grades

Doctor rerum naturalium (Dr. rer. nat.)

vorgelegt von

Timo REUTER

1. Gutachter	Prof. Dr. Philipp CIMIANO Technische Fakultät Universität Bielefeld
2. Gutachter	Prof. Dr. Dr. Lars SCHMIDT-THIEME Wirtschaftsinformatik und Maschinelles Lernen Universität Hildesheim
Prüfungsausschuß	Prof. Dr. Barbara HAMMER Dr.-Ing. Sebastian WREDE

Bielefeld, Februar 2015

Dédié à mes grands-parents

Abstract

Events play a very prominent role in our lifes. Therefore many social media documents describe or are related to some event. However, it is difficult for a human to gather relevant information without any structure in these documents. The organization of social media documents with respect to events thus seems to be a promising approach to better manage and organize the ever-increasing amount of content that is shared using social media applications. It is a challenge to automatize this process so that incoming documents can be assigned to their corresponding event without any user intervention.

In this dissertation we present an event-based stream classification framework that is able to classify a never-ending stream of social media data into a growing and evolving set of events. By doing this, we successfully perform the assignment of a social media item newly uploaded to some social media site to its corresponding event (if it already exists) or create a new event to which future data items can be assigned. We refer to this problem as the *event detection problem* and propose to use machine learning techniques to tackle it.

We successfully address several key challenges that arise in this context: i) handling the data in a stream-based setting, i. e. addressing the challenges arising from the need to process a never-ending stream of data, ii) scaling to the data sizes and rates usually encountered in social media applications, and iii) tackling the new event detection problem, i. e. the problem of determining whether an incoming data item belongs to a new or to an already known event.

We address these challenges through a classification approach allowing us to process the data in one single pass. Furthermore, we include a suitable candidate event retrieval step which retrieves a set of event candidates that the incoming data point is likely to belong to and we include a function trained using machine learning techniques that determines whether the incoming data point belongs to the top-scored candidate or rather to a new event. The performance of our system is maximized using different optimization strategies so that it outperforms many other state-of-the-art approaches.

Further, we extend our framework so that it can be used in a multi-pass setting. Using this approach we show that we can improve the quality of the clustering significantly in comparison to the single-pass approach, while also lowering the computational time by one order of magnitude. We show that this extension can be used in a stream-based setting while reaching the quality of a computationally very expensive offline clustering algorithm.

We prove that our highly efficient approach is capable of successfully clustering a real-world and non-toy dataset by introducing a new dataset consisting of user-contributed images together with associated metadata describing the events they depict. The dataset was already published earlier and is well known in the community. Our single-pass and multi-pass strategies reach an F-measure score of 88.6 % and 93.9 %, respectively. In conclusion, we show that our framework is not only capable of addressing the above mentioned challenging issues but also outperforms other state-of-the-art approaches in terms of quality and scalability.

Acknowledgments

The work presented in this thesis was carried out in the Semantic Computing group, headed by Prof. Dr. Philipp Cimiano, at the Faculty of Technology, Universität Bielefeld. It was partially supported by the Deutsche Forschungsgemeinschaft (DFG), Excellence Cluster 277 "Cognitive Interaction Technology" (CITEC).

During my PhD, there were numerous people that gave me helpful comments, remarks, criticisms, and encouragement. All this support helped me a lot and improved this dissertation further. Therefore I would like to thank every single person for their valuable input.

Especially, I would like to thank my supervisor Philipp Cimiano. It was him, who believed in me and gave me the chance to do this PhD. He had always put a lot of effort guiding me, and I am more than thankful for the chance to benefit from this. He not only knew exactly when it was necessary to push me back on the right path, but always was available when I had questions or needed advice.

I also want to say thank you to my first collaborators from Universität Hildesheim, Lars Schmidt-Thieme, Krisztian Buza, and Lucas Drumond. It was a pleasure to collaborate with you and your ideas and opinions definitively gave me valuable hints to bring my research forward.

Furthermore, I would like to thank Shlomo Geva for the great support during my research stay at the Queensland University of Technology, Australia. He and Chris de Vries helped me a lot to organize the MediaEval Social Event Detection challenge and gave me further ideas on my work.

Special thanks go to my colleagues Christina Unger and Cord Wiljes from my working group for the numerous hours they have spent to proof-read this thesis. I also want to thank Sebastian Walter, Maximilian Panzner, Oliver Beyer, and all the others of my colleagues; you made my journey so enjoyable and provided me with more than professional advice.

I also want to thank my family, especially my mother an father, who always believe in me. They made all the achievements of my life possible by their unlimited support.

Last but not least, I also have to say thank you to my wife Jenny. She gave me the strength and has been on my side during all these years. She contributed a lot and gifted me with her patience for the uncountable number of hours I had to spend making this work possible.

Contents

List of Figures

List of Tables

PART I

Introduction

CHAPTER 1

Introduction

I claim not to have controlled events, but confess plainly that they have controlled me.

— **Abraham Lincoln**
(President and Statesman)

Nowadays, we live in a world in which Internet and social media are taken for granted by a lot of people, especially younger ones. It is well understood that using social media is a great way of getting content published reaching a wider audience. There are many people who express their feelings, share information and media about their lives using social media applications like Facebook, Flickr, YouTube, etc. Young people publicly identify themselves around the content they have posted on the Internet. Even companies are more and more using different types of social media channels to communicate with their customers. The publishing of content has never been easier than today with all the possibilities social media sites are offering on the Internet. Therefore, the data produced using these applications is characterized by an ever-increasing amount of content representing a continuous, never-ending data stream growing at very high rates.

As of September 2014, around 350,000 messages per minute are for example posted via Twitter[1], a popular service for posting short text messages of at most 140 characters to following peers. An image database by Yahoo Inc., Flickr[2], where people are able to upload their pictures to, counted an average of 2,500 pictures uploaded per minute in 2013. While this is already a high value, on Facebook[3], a popular social network site, even more pictures are uploaded: about 250,000 per minute. As people like to share what happens around them, it seems obvious that more and more people are using their digital cameras to take pictures of important and interesting happenings in their life which they regularly upload to social network applications on the Internet. This does not only include happenings that concerns them personally but also unplanned incidents they get involved in.

[1]`http://www.twitter.com`—last accessed 2014-10-04
[2]`http://www.flickr.com`—last accessed 2014-10-04
[3]`http://www.facebook.com`—last accessed 2014-10-04

Many social media sites do not only provide the possibility to share media (e. g. pictures or videos) but additionally enable people who are part of their online community to enrich media items with textual descriptions like a title or comments. As a result, not only the amount of media uploaded to these services is still increasing every year, but also their associated textual metadata.

However, without any structure in these data collections it is difficult for a human to take out relevant information. Another challenge is to combine all relevant information about a certain event. For many people it is highly desirable to browse and search this content in a feasible and, more importantly, in a natural way reflecting human cognitive perception. While social media content still keeps proliferating, techniques for structuring this massive data become crucial.

As events play a prominent role in the world, and the life of a human being is characterized by occurrences and events, many of the information and documents uploaded to social media sites are related to some event. Research by psychologists has found out that the manner of being event-driven is in the nature of humans [WCC07]. Therefore, a categorization of social media documents into events seems to be a promising approach to us for better organizing this huge amount of user-generated content in social media applications.

1.1 Motivating Use Cases

Today, there is an incredible number of possibilities for leisure activities. Unfortunately, most people's spare time does not allow to try all of them; usually the activities have to be chosen wisely in order to organize one's leisure time effectively. As people are socially connected online, the question arises why they should not make use of the event information provided by their friends and others on these social media sites for their personal plannings. In what follows we present three of these use cases. Therefore, let us imagine Alice and Simon, a younger couple.

Case 1—Rare Event Attending: Alice proposes to go out to a small event called "University Dance Fever Night" which is taking place at the city university main hall. Her friend Simon also likes to go out but, as neither Alice nor him did hear about the event before, he is not willing to spend money and time for visiting an unknown niche event which is more than an hour's drive away. For that reason, he tries to search relevant information about the event on the Internet. Contrary to well-known and large events like the Vienna Opera Ball or the Grammy Awards ceremony, about which information and opinions from others are easy to find, it is more difficult to find information about the small event. In this case, searching on a standard Internet search engine for "University Dance Fever Night" will not reveal relevant details as information is not aggregated on one single page but it is spread over several social media sites. It would be desirable for Alice and Simon to find pictures, reviews, and opinions about the "University Dance Fever Night" from the years before which were posted by participants on various social media channels.

Case 2—Finding Event Material: Finally, our couple attended the "University Dance Fever Night" event. They liked the event a lot and the next day Alice and Simon regret that they are not able to share pictures with their friends as Simon did not bring his digital camera because of the fear that it could be stolen. Nevertheless, there were a lot of other people taking pictures of the event and even Alice and Simon were photographed by some unknown persons. Many of these pictures were uploaded and shared on some social media channel. However, as they do not know any of the persons participating in the event, without an appropriate search function, our couple does not find any of these images.

Case 3—Event Summarization and Enrichment: At home, Simon maintains a photo collection of all kinds of events our couple is experiencing. He searches relevant information about each event so that he gets images and meta information from others that he can add to his collection in order to get an event summarization. The intention is to find details which were not observed by the couple but are nevertheless interesting in the future. Unfortunately, this process is not automated. In addition to that, he does not know how certain content on social media channels can be found.

For all previous cases a person desires a technique to find aggregated information from multiple social media sources helping to get an opinion about rare events, to find material about it, and to enrich existing collections. It is desirable to search for a specific event and this search returns all relevant media like pictures and videos but also textual information like facts or opinions. The creation of techniques supporting such a scenario is still challenging. But even though many social media sites support tagging (adding keywords) to better organize the content, the search for specific data items is still demanding. The requirement is not only to find all but, more importantly, relevant information.

Therefore, it is an impending need to better organize and manage this information in a way so that it is well-structured. The structured information then enables us to develop or use current algorithms and methods which facilitate search and navigation of this huge amount of content. This would not only facilitate to satisfy the information need of each individual but also allows new insights for many researchers of how people plan their lives and what is trending.

1.2 Goal and Challenges

> ” *Photos. Life is full of special moments. So is your photo library.*
>
> — **Tim Cook**
> (CEO of Apple Inc.)

We cannot deny the truth about the quotation from Tim Cook. Since the invention of digital cameras and cellphones with camera support, many people possess a personal photo library. We regularly take photos of the moments and events in our life and add them to our photo library. The organization of that library is usually done by hand using one's preferred photo library program. For avid photo shooters this can already be tough from time to time.

Recently, as we already discussed in the motivation, people do not only build and use a personal photo library but desire to enrich it with more content rounding up their collection by adding material from others who attended the same event. This includes also more detailed textual information about the captured moments. For some people this even leads to an almost complete *life-log*. Unfortunately, it is hardly possible to invest the needed time into the organization of such an advanced type of photo library.

The goal is thus to automatize the generation of such a photo library. To this end, this dissertation presents an automatized approach that performs the assignment of a social media item (newly uploaded to some social media site) to its corresponding event (if it already exists) or creates a new event to which future data items can be assigned. We refer to this problem as the *event detection problem* and propose to use machine learning techniques to tackle the challenge of categorizing social media items. In this thesis, we apply our developed framework to captured images with associated textual metadata. Nevertheless, the presented technique is not limited to this kind of social media item but can be used with all kind of items (like videos or tweets) as long as they can be represented in a textual way.

In order to create a suitable event clustering framework for that goal, we have identified different subtasks to be tackled:

- **Feature Selection:** A document (e. g. image, video) is represented by one or more features. These features might be for example time data, imagery data (e. g. pixel or color information), titles, or keywords. It is crucial to get a deeper understanding of how such a document can be efficiently represented by these features.
- **Multi-feature Similarity Metrics:** The representation of the features must allow us to introduce similarity metrics allowing an automatic interpretation of similarity between data items (event and document). To be used in a computer system, we prefer a numeric representation with gradual scale. The chosen features and similarity metrics have to allow us to apply learning algorithms and must also allow an identification of new events.
- **New Event Detection:** The task of new event detection is a classification task. We have to introduce a classification approach allowing a decision if a new data item can be found in the event database or if a new event should be created.
- **Clustering:** We have to introduce a streaming or online clustering approach. The task of the algorithm here is to determine event boundaries and the assignment of new data items to their corresponding event (the event to which this item naturally belongs).

In order to get a working system for event clustering, the above mentioned tasks have to be combined. In addition, there are also several challenges which arise for such a framework as a whole. As the system has to work with a special type of data, we have to tackle different challenges. In detail, they are as follows:

- **Clustering of Large Datasets:** Usual applications in social media have very high upload rates. The data items already available in most applications are often more than a billion items with thousands added every minute. A thoroughly processing of all items is not possible with very limited resources. It is challenging to find algorithms which make a processing of all data possible.

- **Clustering of Continuous Data Streams:** While in usual clustering tasks all data is known beforehand and all data points in the dataset can always be accessed in the preferred order, this is not the case when dealing with stream data. Therefore, the category assignment needs to be fast and efficient. Furthermore, the data must be processed in sequential order and usually in a very little number of passes, in the best case in one single pass.
- **Clustering of Concept Drifting and Changing Time Series Data:** Creating events from documents in a data stream does not end up in a fixed number of categories. Instead, the number of categories is not pre-determinable. In addition, theoretically, there can be an infinite number of categories. Furthermore, the category concepts and types evolve over time; this is known as *concept-drift*. A model must adapt to these concept changes.
- **Noisy Data:** Social media channels are neither proof-read nor is there an instance checking if the content itself is useful at all. In addition, taxonomies or predefined categories are non-existent. It depends on each user how much information is provided, how well-edited the content is, and how properly it is published at all. It is very challenging to distinguish between noise and signal.

In the following sections, we are going to address the above identified challenges and provide more details on all of them.

1.2.1 Clustering of Large Datasets

In machine learning, clustering approaches are often restricted to small datasets. The reason for this usually lies in the processing time for each data item. In many cases the processing time for one single item is already longer than single-figure milliseconds. As a result, the processing of too many data items is thus not possible within a reasonable period of time, even though the algorithm itself works quite effectively. As a result, the datasets used are adjusted to have only a limited number of items. Often the number of clusters is limited, too.

Unfortunately, social media data is massive and the resulting dataset will be huge. In addition, the number of categories created will also be very high. This has a huge impact on how the data has to be processed, i. e., there are different requirements which play an important role. Usually, if datasets are very large, algorithms which can process data in parallel are desirable (see Zhang, Ramakrishnan, and Livny [ZRL96]). As the datasets here are not only large but they are also represented by a continuous stream, using parallelism is not unrestrictedly possible. Nevertheless, scalability is a crucial point and thus several restrictions apply when using such datasets on state-of-the-art computing machines. Consequently, there are several requirements having the large size of the dataset in mind:

- **Memory Usage:** The model as well as the data to be processed is assumed to fit into main memory. Even though many servers include plenty of memory, the fitting of all data into memory is often just impossible. The size of the dataset in a stream data scenario is practically infinite. Thus, it is never possible to fit all data into memory. Relevant data for processing must necessarily be loaded from disk. This process is computationally expensive. Recent database technology can help to organize this more effectively. Indexing of data and smart caching strategies allow a faster retrieval of relevant parts from the dataset.

- **Data Point Representation:** The raw storage of documents and clusters is problematic as searching in raw data is too time-expensive. Therefore, the representation of the documents and event clusters have to be made up in a way supporting a fast retrieval and processing. For example, there are two different ways how a cluster can be represented: either by all the documents being part of that cluster, or by a calculated representative, a so-called *centroid*. Using the latter, less information needs to be stored. The challenge is to balance pros and cons of such a data compression.

- **Candidate Retrieval:** No current computational system is fast enough to compare a new item against all other already seen items in a reasonable time. Fortunately, it is possible to limit this comparison to only a few samples of candidates. This commonly used method is called *sampling*. Sampling is a technique which selects only possible candidates where the probability of a successful merging is above a threshold. This preselection of candidates massively decreases the number of comparisons needed, lowering the time complexity. We have to determine the impact on the performance of such a sampling step. It is challenging to choose good candidates without loosing the true corresponding category.

- **Processing of Each Data Point:** Every document must be scanned and processed at least once. In many clustering scenarios, data points are skipped if they do not fulfill certain criteria. This is commonly known as *shedding*. Data shedding is indeed useful for discarding data points not depicting an event. We concentrate on a system where this distinction has been done beforehand. In our case, it is thus not desired to shed any data item. The challenge is to get an overall system performance fast enough to cluster all data points depicting an event in reasonable time.

1.2.2 Clustering of Continuous Data Streams

In contrast to a fixed set of data where all data is available instantly, a continuous data stream has significantly different properties. To process such a data stream we require a setting that is different to the usually off-line setting used for fixed data sets. In particular, we require an on-line or streaming setting. In this case, each data point will be processed with a small, constant number of passes. Usually, each data point is seen only once.

More formally, the continuous data stream can be imagined as a never-ending stream of data points $x_1, x_2, ...$ processed by a clustering algorithm with k centers in the paradigm shown in Algorithm 1.1.

A data stream can be characterized best as a time-ordered sequence of data items like documents, images, etc. This ordering by time makes continuous data streams require a different treatment

Algorithm 1.1: Stream-based clustering paradigm

```
Input  : A never-ending stream of data points $x_1, x_2, ...$
Output : Clustering with $k$ event clusters

set number of clusters for $k$;

while true do
    get new data point $x$;
    determine best $k \in C$ for $x$;
    update all $k$ in current set $C$;
end
```

preventing us from using different traditional clustering approaches. Here, it is not possible to retrieve data points of the future before they arrive and variations of the ordering are also impossible. Gaber, Zaslavsky, and Krishnaswamy [GZK05] give a brief overview about the techniques and algorithms for fixed datasets that can be applied on data streams. Subsequently, we have to consider some requirements when processing stream data:

- **Number of Passes:** In many clustering approaches, data is stored explicitly, which makes it easy to access all parts of the data at any time, allowing a processing of the data in multiple passes. For a lot of clustering techniques this is the only way to achieve convergence [B+98]. In a setting with a data stream, using multiple passes is not impossible, but the realization needs greater effort and causes some issues. One consequence of a processing in multiple passes is a delay in cluster production; the data points are not assigned immediately but are cached for a certain time. This requires temporal storage of the data. Even though this is possible, there is no possibility at all to see data points coming up in the future. These restrictions enforce an usage of algorithms which are capable of clustering efficiently in already one single pass.

- **Adapting Clustering Model:** The clustering model is the essential part in the clustering process. It must be adapted to all varieties of stream data. One challenge is to be able to detect new events by the current arriving data item. As there is no knowledge about future data items, this decision has to be made instantly with only the data which has been seen so far. The created clusters are of different size and nature. We aim to let the model adapt itself to all kind of events.

- **Efficiency of Clustering:** Using data streams, a certain efficiency is required in order to cluster in real-time. Throughout the clustering process, prediction times usually increase because of an increasing number of target categories. It is challenging to apply appropriate techniques which prevent the exponential growth of the search space so that the clustering process can hold up with the incoming data stream.

1.2.3 Classifying of Concept Drifting Time Series Data

As defined in the paradigm in the last section (see Algorithm 1.1), classical clustering approaches like k-Means require the number of categories to be defined beforehand. This limitation is not acceptable if a full clustering of all documents in target categories is favored. In stream data clustering, the number of categories k is unknown. It is even not possible to estimate the number of categories over a period of time as the growth factor is also unknown. For such an event clustering system, it is crucial to adapt quickly to a changing data distribution. If a new data item arrives and a new concept is detected, another category has to be added to the set of categories.

The distribution of event concepts is often fixed in classical settings. Either the type of events is known or the description of the events to be found are given. If this is, like in our application, not the case, the concepts of the categories are varying and can be of different kind. In time series data, event concepts change quickly over time. Subsequently, new concepts emerge while others fade away. We have no initial information about the concepts in future categories. Our model thus needs to be capable of detecting this drift in concepts on the fly.

1.2.4 Noisy Data

When working with social media data, it is necessary to understand how the data is structured and what kind of data we are facing. The data is coming from a multitude of sources and the amount of data is overwhelming. A lot of this data might just be considered as *noise*. Lovett [Lov11] distinguishes between noise and signal as follows:

- **Signal** is what we are looking for. This is the needed streamlined information which is relevant to conduct the task.
- **Noise** is the matter which does not offer much payout and mainly wastes resources, time, and energy.

The type of data used in a clustering task has different features specifying what is considered as noise and what is not. Regarding the noisiness and its treatment, we clearly have to distinguish between social media data and other data (e. g. news data).

Let us take a look at news article corpora. These articles conform with syntactical, stylistic, and grammatical standards which are required for their publication. In this case, it is also possible to use traditional natural language processing tools to extract or enhance information, as well as for noise suppression as the distinguishability between signal and noise is often apparent. Stop word lists can be applied to the text to filter out irrelevant words and are well-known to reduce the noise. Or part-of-speech tagging can be used to identify certain word forms which are then considered as signal or noise.

In contrast, by its nature, documents from social media do not necessarily conform to any of these standards. Furthermore, they contain only little textual narrative and usually consist only of some keywords, a title, and/or a very short description. This noisiness makes the use of traditional natural language processing tools more difficult. The extraction of named entities or the enhancement of the document's representation with part-of-speech tags is rarely possible. Thus, using social media data, the quality of the text itself is often not useable as an indicator for the distinguishability between noise and signal.

Using traditional natural language tools with the intent to remove noise might not necessarily solve the problem but can also worsen it. During the process of noise removal, valuable information might be eliminated because things regarded as noise in usual news articles might provide valuable information in social media. For example, smoothing a document by the replacement of "noisy" words not appearing in dictionaries (e. g. "fYve" to "five") often remove a special meaning and maybe the only link between two documents.

1.3 Research Contributions of this Dissertation

In this thesis we introduce a framework for online stream clustering of social media documents into their corresponding events. This framework tackles all the challenges described in Section 1.2 and describes effective techniques for new event detection and event clustering. The research in this dissertation advances the understanding for evaluation, effectiveness, and behavior of a classification and clustering problem in social media documents. Our main contribution can be summarized as follows:

Contribution We introduce a complete online system that classifies an incoming stream of documents into the distinct events they describe or are related to. The document is either added to an existing event or, if necessary, a new event is created. It will be shown that this system successfully addresses all above mentioned challenges in an experimental demonstration. We show that the problem is successfully solvable using a single- or two-pass approach. Both approaches strictly adhere to the stream-based nature of the data.

Our online system as a whole combines techniques from different areas. These techniques are interchangeable in the framework allowing a very high flexibility. In detail, we provide contributions in the following areas.

Evaluation of Event Identification

- There is a high need for cluster validity. Therefore, we present a real-world dataset for research in the area of social event identification which is suitable to be used in clustering and classification tasks in that area. The dataset supports the training, testing, and evaluation of such algorithms and frameworks. The usefulness of this publicly available dataset was already shown in several challenges and publications.

- We compare our system to several state-of-the-art systems, showing that our system clearly outperforms most algorithms for event clustering. We show that the here presented algorithm reaches the highest score for streaming algorithms and can compete with results of algorithms interpreting the task as a fixed clustering task.

New Event Detection

- We show that the decision whether a new data item belongs to one of the existing events or to a new event can be taken with reasonable accuracy. This decision is made instantly after seeing a single data item.
- We compare different decision models for new event detection and present suitable features to be used in the context of a stream data setting.

Clustering

- The problem of assigning a new data item to its corresponding event can be modeled successfully as a ranking problem where a preselected number of events is ranked according to how likely the new data point belongs to them. By choosing the top-ranked event, we show that the accuracy of assigning a new data item to its corresponding event is very close to 100%. We discuss the impact of different features on this decision using machine learning techniques.
- We show that the system can indeed scale to large numbers of events by using an appropriate candidate retrieval step which retrieves a set of candidates that the incoming data point is likely to belong to. With this step we avoid scanning all events in the database and thus allow our system to scale to very large numbers of events. We show in particular that retrieving a very little number of candidates yields a reasonable trade-off between scalability and clustering quality.
- Our system is much more efficient than comparable systems, showing a nearly constant processing time independent of how many documents the system has processed so far, a requirement for scaling to much larger number of documents.

1.4 Structure and Outline of this Thesis

In this section we give an overview on how this dissertation is structured. In Chapter 2, we start with detailed information about fundamentals which are important and necessary to understand this work. We explain the categorization idea and how this idea leads to event clustering. In addition we also provide a characterization and definition of what we understand by the term *event*.

In Chapter 3 we discuss foundations and related work. We give an overview of clustering and classification algorithms and take a look at certain aspects which are relevant to use these techniques for our task. We will also take a look at other work that investigated in the area of social media data clustering.

After that, we introduce our event clustering dataset in Chapter 4. The aim of the creation of that dataset is to develop a benchmark for real-world applications. We will present how the dataset has been created and what it can be used for.

In Chapter 5, our framework for social event classification is introduced. We outline the architecture of our system and present the single parts of the framework in more detail. We also introduce our problem of social event detection and classification in more detail and explain how this system can tackle this problem.

Chapter 6 then continues with the experimental setup and results of the clustering of our real-world dataset. We present a setup of the system working with one single-pass. We will show that the system presented is indeed able to tackle the problem and present our results together with several baselines.

We continue with the description of our framework to be used in a multi-pass setting in Chapter 7. The adapted system architecture is shown and we discuss prerequisites which are important in order to use the system in a multi-pass fashion. We show the differences to the single pass system and discuss several strategies which can be used to stick with the stream paradigm.

In Chapter 8 we show the experimental setup and results of our multi-pass system. We not only show how the passes may look but also show that the system scales even better than the one using one single pass while ameliorating the overall performance significantly.

In Chapter 9 we provide additional remarks on our single-pass and multi-pass approaches. In a discussion we will analyze advantages and disadvantages of both approaches, giving examples for different scenarios that show how the different approaches can be used. In addition, we do a comparison with other approaches.

Finally, we conclude this dissertation by a summary in Chapter 10. In an outlook we will furthermore discuss possible extensions and modifications of our framework so that it can be used with different data items and features.

CHAPTER 2

Fundamentals of This Work

This chapter introduces fundamentals which are necessary to understand this work. Notably, we have to make clear what we understand by the terms *clustering* and *event*. First, we start with giving a deeper understanding for the idea of *categorization* and *clustering*, what is understood by this term in different disciplines, and why this idea is needed to cluster objects and events. After that, we survey alternative definitions of the term *event* over a variety of domains and finally give an extended definition of what we understand by an event.

2.1 From the Categorization Idea to Event Clustering: Definition and Development

Categorization can be described as a process of dividing up ideas and objects but also events into parts which are as mutually exclusive as possible. This is a basic cognitive process for humans and is part of our everyday lifes. Rosch and Lloyd [RL78, p. 28] write that, "[...] as an organism, what one wishes to gain from one's categories is a great deal of information about the environment while conserving finite resources as much as possible". We all tend to group objects of the world which we see and capture into distinct categories with a specific purpose. Following Harnad [Har05], *categorization* is the basis for the construction of our world knowledge. This implies that objects and also events are grouped into distinct categories. These categories are thus the result from a process to differentiate, recognize and understand objects and ideas in the real world.

The comprehension of the concept of categorization is fundamental for an understanding of classification and clustering approaches in machine learning. We aim to categorize events according to typical features of members of the event category in question. It is natural to use this idea as a model for a categorization approach for events.

The idea of categorization has received attention in many academic fields. In the next sections we take a deeper look into the development of the categorization idea. In Section 2.1.1, we start with the philosophical or so-called classical view. We then elaborate on the modern view in cognitive psychology, in 2.1.2. Finally, in Section 2.1.3 we build the bridge to our definition of what constitutes the term *clustering*.

2.1.1 Categorization in Philosophy – The Classical View

The categorization concept was discussed deeply in the field of philosophy. Many philosophers tried to answer the question of what are necessary and sufficient conditions for a proper definition of categories.

Already about two and a half millenia ago, the concept of categorization appeared in works of Plato, a hugely important Greek philosopher and mathematician. In his theory of forms and ideas, Plato believes that the material world is only a shadow or poor copy of concepts from the real non-material world. The basis for this is Plato's concept of *hylomorphism*, the theory that substances are forms consisting of material. In his opinion, these forms are constant abstract representations of the things around us and are actually the true basis of reality. This theory is exemplified in the *Allegory of the Cave*, part of book VII from his work *Politeía* (The Republic).

In contrast to Plato, one of his students, Aristotle disagreed with his teacher that ideal forms exist separately from particular things. He argued that forms are merely generalizations. Aristotle developed the basic principles of categorical logic. Following this logic there exist discrete entities which characterize categories by a set of properties. This approach has established the basis for natural taxonomy.

However, it has become clear that categorizing is more complex and that Aristotle's view does not suit too well our understanding of how organisms actually categorize. Following Taylor [MHP11, p. 643ff], the classical view mainly does not take into account what is the real reason for organisms to categorize. They use categorization to manage the countless impressions of the environment. Taylor writes that a categorization of entities per se, is not an useful purpose for an organism. These categories would be too rigid and even if they are well-formed in theory they would not suit human perception and action.

In more recent times, philosophers like Immanuel Kant and Ludwig Wittgenstein also investigated categorization. In his theory of family resemblance, Wittgenstein [Sav11, pp. 31ff] alleged that not all objects falling under one category must share all features but they may be connected by common similarities and/or relationships. As examples he used the terms "language" and "game". They are an example of groups that are connected by family resemblances. In the category "games" there are e. g. "board games", "card games", "ball games"; Wittgenstein challenged us by posing the question whether there is anything common to all.

In newer theories like *prototype theory*, mainly cognitive psychologists try to survey psychological principles of categorizations, i. e. the way people really categorize. The prototype theory differs a lot from the classical Aristotelian theory of categorization [Lak87, pp. 2ff], [Lak82], and the next section will outline its basic ideas.

2.1.2 Categorization in Cognitive Psychology – The Prototype View

In the previous section we presented the philosophical view of categorization. Psychologists do not fully agree with the philosophers' view as their understanding is not necessarily in accordance with human behavior. Not only the limitation to fixed boundaries of the categories, which does not hold true for a valid categorization method in human perception, is a reason to investigate more in this area to overcome the limitations, but also the question why categorizations are made at all is of interest.

Rosch and Lloyd [RL78], specialized in cognitive psychology, used Wittgenstein's theories as a basis to find out that humans use representative prototypes as an orientation for categorizing objects instead of using abstract criteria and thus coined *prototype theory*. While Wittgenstein did not propose that some objects are better examples for a prototype than other, Rosch argued that the distance of a specific instance to the prototype varies.

Instead of an equal status within a category with clear boundaries, there are somewhat overlapping categories with better and less good examples. Following that theory, it is possible to specify gradual membership to a category instead of a hard binary membership. In the cognitive approach it is well-accepted that natural categories tend to be fuzzy at the boundaries. It is more or less a matter of the point of view and experiences of a person to decide on the grade of membership for an object [Ros75].

Taylor [MHP11, p. 652] argued that categories are defined with respect to a “best example”. Rosch strengthens that hypothesis as she states that “much actual learning of semantic reference [...] may occur through generalization from focal exemplars” [Hei71]. She also wrote that “another way to achieve separateness and clarity of actually continuous categories is by conceiving of each category in terms of its clear cases rather than its boundaries” [RL78, p. 35–36]. While Rosch and Lloyd [RL78] could be interpreted in a way that every category is represented by exactly one best example, taking a look at work by Rosch and Mervis [RM75] clarifies that the generalization from several good examples as a summary representation was meant.

The following quote from Rosch [RL78, p. 30] leads Taylor [MHP11, p. 653] understand a category as a *set of weighted attributes*:

> “[C]ategories tend to become defined in terms of prototypes or prototypical instances that contain the attributes most representative of items inside and least representative of items outside the category.”

Following Taylor [MHP11], category members neither need to share all attributes, nor must an attribute be shared by all category members. This follows Wittgenstein's idea of family resemblance [Sav11] in which attributes intersect.

An alternative approach is to construct a category as a collection of instances. Smith and Medin [SM81] call this the *exemplar view*. They argue that there is no need to define properties if different exemplars of a category do not need to share any of them. Actually, in this theory, there is no generalization process, instead one memorizes a set of objects belonging to the meant category. Looking up if something belongs to that category consists of consulting the memory very quickly to see which things it is most similar to [Mur02, p. 49].

Independent of which view one adopts, Rosch and Lloyd [RL78] showed also that there are varying levels of abstraction. Therefore, humans categorize in a hierarchical way. They argue that there exists a basic level that is privileged in perception; actually, this is the level with the relatively greatest amount of information. These basic levels are preferred by adults in a neutral setting clearly showing that these tend to be the concepts learned earliest by young children.

2.1.3 Event Clustering Characterization

While the theories described above are mainly about the categorization of things, humans also categorize moments of their lives into events [ZT01; Nei86; BS98]. In artificial intelligence and machine learning, there is the aim to imitate that natural cognitive process by constructing and learning a model from data rather than rely only on programmed instructions.

The categorization theories presented earlier lead us to the task of *clustering*, defined as follows:

> **Definition 2.1.1 — Clustering** Clustering is the task of categorizing information in a way so that objects in the same *cluster* are more similar to each other than to those in all other clusters.

Clustering is not a special algorithm but a common technique used in various fields like machine learning, pattern recognition, and information retrieval for solving the categorization task. The goal of clustering can thus be accomplished using different algorithms. This implies also that the data needed for the processing as well as the resulting clusters will be divergent depending on the used methodology.

Looking back at two of the theories described above, the *set of weighted attributes* theory and the *exemplar view* theory, we see that there are already two very different approaches which can be mapped to different clustering strategies. According to the first theory, the clusters are represented by only one, the best, representative, e. g. a centroid. This centroid might be the mean of features from all objects inside the cluster or the object which has been identified (in whatever way) to be the most centered. To query the distance to that cluster, only that centroid is used. In contrast to that, according to the second theory, all clusters are represented by all objects inside the cluster. If we want to get the distance of an object to that cluster, it is necessary to calculate the distance to all objects inside it and then to aggregate over all those distances.

Summarizing, we intend to develop an approach using machine learning methods for event clustering which is capable of clustering documents in a way that resembles human cognitive perception. The challenge is to automatize this process of event clustering so that incoming documents can be assigned to their corresponding event without any user intervention.

2.2 Characterization of an Event

In the previous section we gave an overview of what we understand by the terms *categorization* and *clustering*. In addition to that, it is also essential to comprehend what exactly can be understood by the term *event* before we can develop a system for clustering events. Taking a look into a dictionary[1], we find three main definitions. Accordingly, an event is:

1. "something that happens or is regarded as happening; an occurrence, especially one of some importance"
2. "the outcome, issue, or result of anything"
3. "something that occurs in a certain place during a particular interval of time"

Looking up the etymology of the word also gives us a hint about the meaning of the word. Its origins are found in the Latin language. The word *event* is derived from *ēventus* which has the meaning of *occurrence*, *accident*, *fortune*, *issue* as well as the past particle stem of *ēvenire* with its meaning *to come out*, *to happen*, and *result*.

As we see, neither the dictionary nor the etymology help us to find only one true definition of what an event is. Instead, there are several alternative definitions. Similar to the debates about the categorization idea, the characterization of the term *event* has received attention in humanities like philosophy and psychology, too. While one might catch a basic idea of what an event is by the definitions above, it is still interesting to take a deeper look into other fields to get an idea about how humans perceive events. Not only researchers but also individuals will disagree largely on what exactly constitutes an event. In 1963, researchers were already aware of this kind of human behavior but did not have an understanding of the ongoing cognitive processes:

> "Temporal aspects of behavior are among the most compelling in experience and among the most easily measured of all of behavior's unnumbered characteristics. Despite the saliency of the time dimension however, little is known about the actual arrangement of behavior along its temporal axis."—Barker [Bar63]

The group behind the ontology DOLCE [Gan+02] defines two disjoint classes after the idea from Lewis [Lew86] where most phenomena of reality fall into *endurants* and *perdurants*. The first, enduring particulars[2] are persisting entities existing in time which necessarily lack proper temporal parts and are necessarily wholly present at all times of their existence. It is rather easy to see that everything falling into the category of endurants can *not* be an event. There are many examples: physical objects (e. g. stones, trees, humans) and psychological objects like ideas, concepts and goals [SZ08, p. 4]. They all have in common that there is no temporal reference involved. The information we gain from this is that events have to include the time factor. This leads us to the class of perdurants. Perduring particulars are persisting entities that happen in time. They have proper temporal parts, and such a perdurant can happen at some point in time. The duration of the happening does not matter and the temporal scales may span from milliseconds to millenia. Examples are knocking on the door or the creation of a canyon.

[1] `http://dictionary.reference.com/browse/event`—last accessed 2014-09-16
[2] A particular is a concrete entity existing in space and time.

In contrast, in the field of physics, an event is simply defined as a physical occurrence located at a specific point in space and time (e. g. a leaf falling down from a tree). Unfortunately, the definition is more complicated in other fields. In the following sections we will discuss the views of other fields like philosophy and psychology, but also in recent computer science, i. e. the areas of machine learning and information retrieval, and try to give a survey over a variety of domains. We finally connect these approaches and theories to our definition of an event adapted to our problem definition.

2.2.1 Event Definition in Philosophy

Metaphysicians have a great desire to understand the basic nature of things that happen. More generally they want to understand if events constitute a basic ontological category or if the human concept of an event is only a way of organizing elements [Ben02].

Contemporary philosophers started with an analysis of *actions*. Actions, that can be said, are actually atomic components of events. A definition of a basic action is provided by Danto [Dan63]: "B is a basic action of a if and only if (i) B is an action and (ii) whenever a performs B, there is no other action A performed by a such that B is caused by A". Even if this leads into the right direction of what an event could be, this is problematic in the way that an *action* is not synonymous to an *event*. An example for a basic action is "moving one's hand", a counterexample is "pointing with the hand". These concepts of actions are only a foundation for the larger construct *event*.

In philosophy, there exist also approaches to characterize not only actions but also events. Until now, there has been no universal definition in that area, but there exist several more or less concurring proposals. Most notably, there are the following four approaches to be mentioned:

1. Kim's Property-Exemplification Account of Events [Kim69; Kim73]
2. Davidson's and Lemmon's Theory of Events [Lem67; Dav69; Dav80]
3. Lewis' Theory [Lew86]
4. Quine's Theory [Qui85]

In the following we describe these approaches in more detail, starting with Kim's theory. All of the theories have strengths and weaknesses with respect to our purpose.

Kim's Property-Exemplification Account of Events

Kim [Kim69] does not directly provide a definition of an event but provides a general and fundamental basis for the creation of a definition. According to him, the term *event* implies change of a substance—following Kim, a substance can be any kind of object with no particular philosophical doctrine. This change requires the acquisition of a property which an object did not have before [Kim76]. Kim sees events as structured and constituted by three elements: i) an object or objects x, ii) a property P, and iii) a time point or range t. He defines an unique event by an expression of the form $[x, P, t]$ with two conditions:

- $[x, P, t]$ exists in case the substance x has the property P at time t (existence condition)
- $[x, P, t] = [y, Q, t']$ if $x = y, P = Q, t = t'$ (identity condition)

> "The theory states that just in case a substance x has property P at t, there is an event whose constitutive object is x, whose constitutive property is P, and whose time of occurrence is t (the existence condition), and that events are identical just in case they have the same constitutive property, object, and time (the identity condition)."—[Kim76]

In addition, he theorized about events that a) they are non-repeatable and unchangeable particulars including not only changes and states but also conditions of the event, b) they have a spatio-temporal location, and c) the constitutive property makes the event individual.

Davidson's and Lemmon's Theories of Events

Lemmon [Lem67] and Davidson [Dav69] agreed that events are understood as logical particulars, i. e. non-repeatable occurrences. They are considered as constants which can be used in first-order predicate logic. The difference between an event and an object is grounded by the view that events are occurrences while objects are definitively not. Davidson explains that an object remains the same through changes, but an event changes in an object or objects [Dav80].

Davidson and Lemmon have extended the theory of events by two conditions: i) the causal criterion and ii) the spatio-temporal criterion. Both authors agree in the causal criterion on that events are identical if they have the same cause and effect. For the spatio-temporal criterion Lemmon proposed that events are also identical if they occur in the same space and time. Davidson later rejected that criterion and gave the following example: "If a metal ball becomes warmer during a certain minute, and during the same minute rotates through 35 degrees, must we say that these are the same event?" [Dav80, p. 178].

Lewis' Theory

Lewis favors a more opportunistic theory telling us which entities are formally acceptable as events. The essence of Lewis' idea for the theory of events is that an event is a property of spatio-temporal regions [Lew86]. In Lewis' ontological scheme, properties (including events) are not basic. Lewis states: "Events fall in with the properties; for I see no reason to distinguish between an event and the property of being a spatio-temporal region, of this or another world, wherein that event occurs. [...] An event that actually occurs, then, is a set that includes exactly one this-worldly region." [Lew86, p. 95]. Lewis proposes the following condition for some entity e to be an event:

> "e is an event only if it is a class of spatio-temporal regions, both thisworldly (assuming it occurs in the actual world) and otherworldly."

In his understanding, any instance of that event class actually "occurs", all others (including the event itself) do not occur.

Quine's Theory

In comparison to the other approaches, Quine [Qui85] made a very simple assumption of what events are. He argued that events should be regarded as regions of space and time. He elaborated further that objects are usually a family of bounded space-time regions, like events. Therefore, as a consequence, following his theory, events could be simply regarded as objects. Thus, there would not be the need to make any difference in the assumptions about both.

2.2.2 Event Definition in Cognition and Psychology

In psychology the assumption is that information about events is stored in a memory system that consists of episodes collected during life, the so-called *autobiographical memory*. This memory consists of a combination of *episodic* and *semantic memory* [WCC07]. Following Tulving [Tul72] and Williams, Conway, and Cohen [WCC07], episodic memory stores information about episodes and events of personal experiences that have been experienced at a certain time and place. Semantic memory instead stores general knowledge about the world. In short, we can define an event or episode in psychology as follows:

> “An *event* is the empirical knowledge in a brain of an individual about happenings of situations or an occurrence of a situation.”

or following Zacks and Tversky [ZT01]:

> “A *event* is a segment of time at a given location that is conceived by an observer to have a beginning and an end.”

The in-depth discussions in psychology are on-going in the areas of cognition and development. While the definitions of what constitutes an event and action in philosophy provides a basis, psychologists aim to come to an understanding how humans perceive and think about anything that happens [SZ08].

In the area of psychology, common questions concern the differences in categorization between events and objects, how events are perceived and segmented, and how events are remembered and thus represented. We address these questions in the following paragraphs.

Contrasting Juxtaposition of Events and Objects

In earlier research, it was not known if there is a great difference in categorization between events and objects [Bar63]. Newtson et al. [New+87] and Zacks and Tversky [ZT01] used the proposal of Quine [Qui85] as an opportunity to investigate in this direction.

Following Zacks and Tversky [ZT01], humans categorize events in the same way they categorize objects (see also section 2.1.2). As proposed in the philosophical approach of Quine, psychologists like Newtson et al. [New+87] and Zacks and Tversky [ZT01] conducted studies that support that approach. These studies finally led to the knowledge that events can be put at the same level as objects. For us, this is an indication that categorization methods used for object categorization can be used without modification for event clustering.

Another analogy of events to objects is that they consist of parts. Zacks elaborates further that these relations produce two forms of hierarchical organization: partonomies (also known as meronomies) and taxonomies. The difference between both hierarchies is that the first one bases its categorization on part-whole relationships and the latter on discrete sets. The interpretation of edges in partonomies is "part of"; in taxonomies, it is "kind of". Zacks and Tversky [ZT01] even assume a combination of both organization forms in human perception. They found out that events are characterized mostly at the basic level of the hierarchy.

In contrast to the positive analogies above, there is one substantial difference between events and objects: events are necessarily ephemeral[3] which is not the case for objects [ZT01; McC93]. This seems reasonable as it is indeed possible to recognize a particular object but it is impossible to experience the same event more than once, e. g. every Christmas is celebrated only once. Nevertheless, it might be perceptually categorized as a type of Christmas celebration. Events can be perceptually categorized, but they cannot perceptually identified.

Segmentating and Perceiving Events

Newtson and Engquist [NE76] and Boltz [Bol92] showed in their experiments that people agree largely on so-called breakpoints in short-range action and long-range events. They even show that these breakpoints have a tendency to correlate with the moment when physical features of actions are changing [NEB77]. Therefore, it is allowed to assume that people have a clear foundation for event boundaries.

It can also be assumed that people actively modulate the event segmentation level. for example, people are able to divide events at other levels if they are directly asked to do so [LSR88]. Zacks and Tversky [ZT01] claim that the spontaneous variation in segmentation level is needed due to the aim of the brain to minimize resource consumption. For a human being, it is important to maintain a coherent understanding of the surrounding happenings. If the temporal piece is too coarse, the brain shifts to a finer segmentation. Zacks and Tversky [ZT01] go further and allege that if events are viewed on longer time-scales, they become less tied to physical activity.

While Newtson et al. [New+87] deny that humans encode activity in partonomic hierarchies, Zacks, Tversky, and Iyer [ZTI01] more recently found in experiments that this is actually the case.

For the segmentation and perception of events, Zacks and Tversky [ZT01] highlight three important points:

1. On-line segmentation techniques can be used to study temporal parts of events.

2. Events seem to be perceived in a partonomic structure.

[3]lasting only a period of time

3. Events have a causal perceptual structure and a partonomic perceptual structure. Both might be highly correlated.

Remembering and Representing Events – Autobiographical Knowledge Base

Perceptions, actions and events are structured by repeated exposures to event schemata collected through experience. Following Zacks and Tversky [ZT01], the schemata provide for i) variable binding (allows goals and roles), ii) embedding (allows partonomic structure), iii) varied levels of abstraction (allows taxonomic structure), and iv) they represent knowledge (allows adaptive, probabilistic perception). Recognizing an event as a certain type of event category is done by matching it to schemata stored in memory. If information is missing in perception, it is filled in by reference to retrieved schema (see also Brown and Schopflocher [BS98]).

Conway and Pleydell-Pearce [CP00] conclude from older research that the autobiographical memory is structured into different areas: *lifetime periods*, *general events*, and *event-specific knowledge*. Examples for lifetime periods are “When I was young” or “When I was at school”. They represent general knowledge about several things like significant others, common locations, actions, activities, plans, and goals. Contrary to lifetime periods, general events are more specific and more heterogeneous. Both repeated and single events are included in this area [Bar88]. They also can represent sets of associated events (a series of memories) [Rob92]. General events are grouped into clusters and missing information is recalled from memory; this is in accordance to Zacks and Tversky [ZT01]. Event-specific knowledge comprises high levels of detail about certain events (like images). Following Conway [Con05b], the combination of these areas are organized in a hierarchy within the autobiographical knowledge base and make up everyone’s life story.

There is a differentiation between *specific* and *generic* memories in the literature [WCC07]. This is reasonable as one might remember a specific, particular event like today’s interest group meeting or just think about a generic group meeting.

It is important to recognize that memories fade. As a consequence, over time, events are remembered in a less fine-grained fashion [KZ08]. While one remembers every single day directly after a holiday, this fine-granularity will be lost after some time.

2.2.3 Events in Recent Literature of Machine Learning and Information Retrieval

In various tasks in the area of machine learning and information retrieval, there are different scenarios where definitions of the term *event* are necessary. Not only in the very common task of event and topic detection in news feeds [All+98], but also in several other domains like videos, insider trading, surveillance and crime detection, remote sensors, as well as social media platforms, there are some kind of events involved. It applies also to the extraction of events from articles and documents. In this section we give a survey about the main tasks in the area and show different characterizations and definition of what is understood by the term *event* here.

Event Detection in News and Stories

Talking about events in information retrieval, the area of event detection in news and stories is one of the most prominent ones. This research area is commonly known as *topic detection and tracking (TDT)*. A lot of researchers have carried out experiments to detect events in news stories. The main question which arises here is how an event can be characterized.

Beginning in 1996, the Defense Advanced Research Projects Agency (DARPA) pushed the research in TDT a lot. In a pilot study, Allan et al. [All+98] surveyed the field of TDT. Its purpose was to find and follow new events in streams of broadcast news stories. They identified three major tasks: i) the segmentation of stream data into distinct news stories, ii) the identification of new news stories when they first occur, and iii) finding following new stories given a small number of examples of a certain event. The annotated corpus for the studies was produced by the Journal of Graphics Institute with respect to 25 target events that have been defined according to the TDT guidelines.

In the study of Allan et al. [All+98], the notion of *topic* was made explicit by its interpretation of *event*. Their original definition of the term *event* is as follows:

> "[S]ome unique thing that happens at some point in time."

It is notable that this first definition just includes a temporal reference. They elaborate further that they differentiate between events (e. g. the eruption of a certain volcano) and classes of events (e. g. volcanic eruptions). They also allow events to be unexpected as well as expected.

While Allan, Papka, and Lavrenko [APL98] and Yang, Pierce, and Carbonell [YPC98] already altered the definition to include a spatio-temporal instead of only a temporal reference, the TDT initiative [All02] went further and offered a more specific version of the definition in a later publication:

> "[A] specific thing that happens at a specific time and place along with all necessary preconditions and unavoidable consequences."

In addition, Allan [All02, p. 42] define a *topic* as "an event or activity, along with all directly related events and activities". In TDT, these topics are predefined; for a full list refer to Allan [All02, p. 43]. Both above mentioned definitions clearly define the boundaries of events as used in the TDT tasks. However, these definitions only cover event types used in news stories. Furthermore, Makkonen [Mak03] has identified some flaws regarding necessary preconditions and unavoidable consequences for events. Nevertheless, newer publications in new and popular event detection and tracking still agree with that definition [ZZW07; Lin+10].

Information Extraction

The aim in event extraction is to identify and extract events in text. One use case for this method is in the area of biotechnology and health [Kim+09] where researchers want to detect events from papers and literature to provide overviews of events together with entities and relations. Definitions of an event in these areas are very specific to each area and are not useful to be considered in our context.

Event extraction has also been used on news sources. Gaizauskas and Humphreys [GH97] used a semantic network for identifying events already in 1997. Later, as part of the Language Resources and Evaluation Conference (LREC), the Automatic Content Extraction (ACE) program has been introduced in 1999. The objective in ACE is to infer events, entities and relations from human language, notably news. In the beginning, ACE just detected and tracked entities and relations. Starting in 2004, events were added to the tasks, too [Dod+04].

Definitions about the term event do not vary over publications as the task defines it in the ACE guidelines [Con05a]:

> "An Event is a specific occurrence involving participants. An Event is something that happens. An Event can frequently be described as a change of state."

Event Detection in Multimedia

The approach in event detection in multimedia like video and audio clips is different from the other approaches as there are no textual features involved. Lew et al. [Lew+06] report that first tries for boundary detection of events in videos were already made in the 1990s. It became more popular in recent years as machines became capable of handling video data. While first research was already done in the TREC Video Retrieval Evaluation (TRECVID) in 2008, a special task for event detection was introduced in 2010 with the TRECVID Multimedia Event Detection (MED).

TRECVID MED wants to encourage researchers to develop techniques which allow a search of user-defined events in multimedia collections via a metadata store. The definition of what is understood by the term *event* according to the TRECVID MED 2014 evaluation plan [Tra14] is as follows:

> "An "event" is a complex activity occurring at a specific place and time involving people interacting with other people and/or objects. An event consists of a number of human actions, processes, and activities that are loosely or tightly organized and that have temporal and semantic relationships to the overarching activity. All events are directly observable."

In these definitions the terms "actions" and "processes" are not well explained. But, as the desired events are provided by the task maintainers, the definitions together with examples of the desired events shall be enough to grasp the general idea.

Event Detection and Identification in Social Media

In this dissertation we are working with social media data. Therefore, it is obvious to take a look at event definitions in similar work in this area. As we have already seen in other study fields, there is no need to define an all-universal definition. In the style of philosophical thought experiment regarding observation and knowledge of reality, someone on the Internet came up with the following quote:

> “If a tree falls in the forest and nobody tweets about it, did the tree or the forest ever exist, and did the event ever happen?”—Unknown

Rethinking of the quotation above, it is undeniable that at least one human individual has to be involved in the event and furthermore someone has to talk about the happening. In early publications, the definition from the TDT task has been adopted (see above) [BNG09]. Later, Becker et al. [Bec+12] refined their definition to be more specific:

> “[W]e define an event as a real-world occurrence e with (1) an associated time period T_e and (2) a time-ordered stream of social media documents D_e discussing the occurrence and published during time T_e.”

There has been also similar work done in the MediaEval Benchmark for Multimedia Evaluation (MediaEval) Social Event Detection (SED) task [Lar+13]. In the task description of SED, researchers are asked to discover and describe social events in a collection of social media content. Over the years there were different tasks to be done [Pet+14]. While there was first the aim to find just event categories (e. g. soccer matches), it evolved to the question if a certain instance of an event (e. g. the soccer match between France and Spain in September 2014) can be found. Here, the definition of the term event is as follows [Reu+13]:

> “Social events are events that are planned by people, attended by people and for which the social multimedia are also captured by people.”

Event though the definitions are somewhat different, we see nevertheless that they strongly adhere to the social media aspect.

2.2.4 Discussion and Definition

In the preceding sections, we presented numerous approaches from different researchers of various fields of study who wrestled with definitions of the term *event*. Theories from philosophy provide a very good grounding and set the minimal requirements for definitions in other disciplines. As an essence, we learn that the majority agrees upon the theory that an event consists of some activity at some time and place which is conducted by some agent. This theory is also valid for our understanding of an event.

Regarding the experiments and definition in the area of psychology, we agree that an event is not some invented artificial construct but a fundamental concept which is part of every cognitive system. It is thus obvious for us to build an event clustering system which strongly adheres to that natural concept in human perception as the system shall categorize items to events understood by the users. As a consequence, a definition must include the ideas and findings from psychologists.

Finally, in addition to the other arguments, we also like to include a hint to social media. When working with social media, it is obvious that not all types of events are included but only the ones people are talking about. This factor also brings out that it is necessary that persons are involved.

In the following we thus present our definition for the term *event*. Furthermore, we define the identity condition and the event granularity and hierarchy. All these definitions are in line with the discussed premises.

Event Definition

> **Definition 2.2.1 — Event** An *event* is an occurrence or happening at a segment of time at a given (real-world) location that is conceived by several observers to have a beginning and an end, and that is planned, carried out, and talked about by humans.

Event Identity Condition

> **Definition 2.2.2 — Event Identity** Two events are the same only if they are the same occurrence or happening at the same segment of time at the same location.

Event Granularity and Hierarchy

> **Definition 2.2.3** Events with the above definition can be of different granularity. Usually, there is a hierarchy of events, meaning that there are higher level events that comprise lower level events. The farther back an event is in time, the more important are the higher level events.

CHAPTER 3

Foundations and Related Work

In this chapter we review the literature related to this dissertation and describe foundations that are directly relevant. As we have stated before, our problem of classifying documents from social media applications into distinct events can be tackled using machine learning algorithms. These algorithms play a key role in successfully designing appropriate learn functions for the task. Therefore, the subject of this work is to explore machine learning methods for the creation of an approach to classify stream data into event categories. Regarding the data to be processed and the task, we mainly have to deal with two widely used methods in machine learning for the analysis of data: *classification* and *clustering* [Mit99]. In the following we will discuss machine learning in general and the application of machine learning techniques for stream classification in particular.

Regarding machine learning, the term *learning* refers to systems that have the ability to learn to solve a task by example or observation. In order to have such an ability, these systems need an appropriate learning method. Cherkassky and Mulier [CM07] define a learning method as follows:

> "A learning method is an algorithm [...] that estimates an unknown mapping (dependency) between a system's inputs and outputs from available data, namely from known (input, output) samples. Once such a dependency has been accurately estimated, it can be used for prediction of future system outputs from the known input values"—[CM07]

The mapping (or dependency) Cherkassky and Mulier [CM07] mention can be expressed as a function $h(x) : X \to Y$ mapping an input space X to an output space Y. The function $h(x)$ as well as the input and output differ depending on the actual task. In machine learning, the aim of the learning phase is to train or learn the function $h(x)$ so that it fits a training dataset X that is represented by individual observations $\{x^{(1)}, ..., x^{(m)}\}$. The final goal is to learn a function $h(x)$ which can be used for the successful mapping of yet unknown observations x to accurate outputs $y \in Y$, minimizing the risk of misclassifying.

Traditionally, machine learning approaches can be categorized into two different groups: *unsupervised* and *supervised* learning approaches. Unsupervised approaches are, for example, clustering algorithms like k-Means or Expectation-Maximization (EM). Other examples are dimensionality reduction algorithms like Linear Discriminant Analysis (LDA) or Principal Component Analysis (PCA). To train an unsupervised model, only $\{x^{(1)}, ..., x^{(m)}\}$ are used for the training without any information about a designated outcome $y^{(i)}$. Usually, these algorithms are used if the designated outcome is not known or difficult to produce. If the designated outcome $y^{(i)}$ for an observation $x^{(i)}$ is known, a supervised approach can be used. Supervised algorithms can be learned using data tuples in the form $(x^{(i)}, y^{(i)})$.

The most relevant machine learning approaches for our task are:

- **Classification**: Classification aims at identifying which of a set of given categories a yet unseen data point should be assigned to. This decision is made on the basis of past observations where the category membership is known. This is a supervised strategy.
- **Clustering**: This approach aims at discovering clusters in a given set of data points according to some cluster criterion. This criterion could be a similarity or distance. Usually, this process is unsupervised.

The difference between these two methods is illustrated in Figure 3.1 based on an overview by Kuncheva [Kun04].

An important question and a prerequisite to use any machine learning technique is which type of data can be used and how it can be acquired. Typically, datasets consist of data points of human-created or natural artifacts. This might be images, news articles, or sensor data. No other information about the data points is included. For example, there is no further description of how these data points are related, etc. Such a collection of data is called *unlabeled*.

If the unlabeled data is augmented with some information about relations between data points, it is called *labeled* data. This information might be a classification, a distance between data points, or a meaningful label of the class of category the data point is about. These labels are often obtained by asking humans to judge and classify the data. For example, in the case of image clustering, this is information about images forming a group or belonging to the same category. If human assigned categories exist for a dataset, it is often referred to as a *gold standard*. In our case, this gold standard includes example data where the similarity between data points is known.

As such a gold standard (annotated data) exists for our problem, we are able to frame the task as a classification or clustering problem. In the following, we will therefore discuss these two related techniques. After that we discuss an extension of clustering algorithms incorporating supervision by the use of background knowledge (Section 3.4). In Section 3.5 we discuss different possibilities on how these techniques can be applied to large-scale datasets. Section 3.6 then presents different approaches in the field of new event detection. Finally, we conclude the chapter in Section 3.7 with an overview of directly related work surveying the field of social media applications and social event detection.

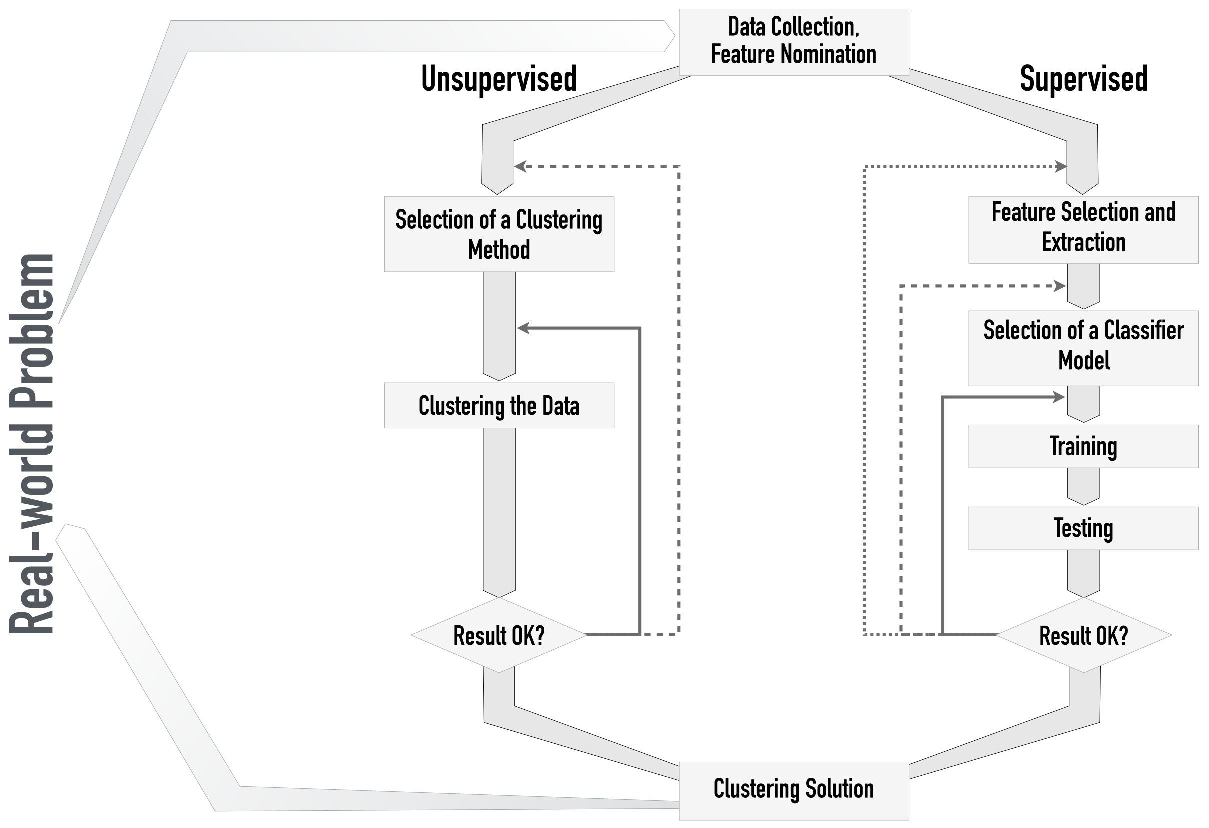

Real-world Problem
Data Collection, Feature Nomination
Unsupervised
Supervised
Selection of a Clustering Method
Clustering the Data
Result OK?
Feature Selection and Extraction
Selection of a Classifier Model
Training
Testing
Result OK?
Clustering Solution

Figure 3.1.: Difference between unsupervised and supervised learning strategies according to Kuncheva [Kun04]

3.1 Classification

Classification denotes a method to assign items to categories or classes. The problem is to identify the correct category for a new data point that has not been seen so far. Typically, we have the goal to learn a function f that maps an input vector X to an output vector Y:

$$f : X \to Y \tag{3.1}$$

The function f can be learned using suitable training data in the form of pairs $(x^{(i)}, y^{(i)})$. If we assume that this data is categorized according to a multi-dimensional feature vector so that $x^{(n)} \in \mathbb{R}^d$, the function to be learned is $f : \mathbb{R}^d \to Y$.

The function learned from the training data is only an approximation of the function f that we want to learn but do not know. Therefore, we denote the learned function as $\hat{f}$. Our goal is to minimize the error on the outcome when predicting unseen data items. In other words, we aim to minimize the risk taking a wrong decision. This risk minimization can be achieved by keeping the average error on the training data at a minimum.

Therefore, a function is needed that quantifies the output error. Such a function *err* differs depending on the output type and on whether the output is continuous. If the output is linear, the mean squared error can be used:

$$err(X) = \frac{1}{n}\sum_{i=1}^{n}(f(x_n) - \hat{f}(x_n))^2 \tag{3.2}$$

If the output is discrete, the following binary error function can be used:

$$err(x) = \begin{cases} 0, & \text{if } f(x) = \hat{f}(x) \\ 1, & \text{otherwise} \end{cases} \tag{3.3}$$

The learning algorithm tries to minimize the error while searching for a suitable function $\hat{f}$. The quality of the learned function depends on the ability of how well it can generalize to unseen data points. There are numerous examples for actual algorithms that can be used for a supervised classification: Decision Trees, Support Vector Machines, Perceptrons, etc. All these algorithms are just different model classes providing a learning function. A frequent model is a linear function:

$$f(x) = w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_n \cdot x_n = w^T x \tag{3.4}$$

The objective of the learning function is to parametrize the model class so that the error is minimized. Thus, supervised classification can also be seen as a smart search within the space of all possible functions.

3.2 Clustering

The field of clustering is broad and the number of different algorithms is high. In a position paper by Estivill-Castro [Est02], the author argues that "clustering is in the eye of the beholder". Following Estivill-Castro, any clustering of a dataset, whether produced by a human or algorithm, is a hypothesis for data groupings which is selected among a set of possibilities structured in some way. This becomes a data model which might potentially be a mechanism to classify instances of that data. The preference of one hypothesis over another is done using some *clustering criterion*. This criterion can be seen as the mathematical formulation of the inductive principle [Est02; X+05; Nal+08]. Estivill-Castro [Est02] states that models are the structures to represent clusters given a dataset, and the induction principle is selecting a "best fit" model.

In contrast to the classification task, clustering is generally regarded as an unsupervised learning problem [JMF99]. It is fundamental for the analysis of data of which the structure is not known. The basic task is to discover similarities and structures within a collection of data points.

Clustering has been studied extensively in the literature and has important applications in many areas. This includes data mining [HK06], vector quantization [Ste56; Mac+67], statistical data analysis [BR93], and others. A lot of disciplines have introduced different kinds of clustering methods and techniques to build classifications for structuring knowledge.

Based on work by Fayyad et al. [Fay+96] and Halkidi, Batistakis, and Vazirgiannis [HBV01], the whole process of clustering can be summarized by the following steps:

1. **Feature Selection:** Selection of the features based on which the clustering is performed. The data must be preprocessed so that as much information as possible is encoded to reach the clustering goal.

2. **Similarity Computation:** Determines how similar or dissimilar data points are. It is desired that all features contribute equally to the similarity measure, because features dominating might produce an undesired clustering. This is described in more detail in Section 3.3.

3. **Clustering Criterion:** A clustering criterion has to be defined in order to determine how how good a clustering is. This is done by means of some rules or a cost function.

4. **Clustering Algorithm:** An appropriate clustering algorithm which can be used to optimize the clustering criterion has to be chosen or developed. It should fit the chosen dataset.

5. **Result Validation:** The quality and the correctness of the resulting clustering have to be evaluated. These results can then be used to refine the whole clustering process.

One of the simplest clustering algorithm is k-Means. This algorithm aims to partition n observations $x_1, ..., x_n$ into k clusters $C = \{C_1, ..., C_k\}$. As similarity measure it usually uses the Euclidian distance. The clustering criterion serves to minimize the squared distances within the clusters. Therefore, the clustering criterion optimized by k-Means is:

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{3.5}$$

Often, k-Means employs Lloyd's algorithm. Given an initial (random) set of k centroids $c_1, ..., c_k$, this algorithm alternates between two following steps until convergence is reached.

- **Assignment**: At each iteration t, each data point is assigned to the centroid whose sum of squares (squared Euclidan distance) is minimal, such that:

$$S_i = \{x_p : \|x_p - c_i^{(t)}\|^2 \leq \|x_p - c_j^{(t)}\|^2 \quad \forall j, 1 \leq j \leq k\}, \tag{3.6}$$

- **Update**: The centroid is recomputed so that it represents the mean of all data points belonging to the cluster:

$$c_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{3.7}$$

A convergence has been reached when the assignments no longer change, such that a (local) optimum is found. A drawback is that there is no guarantee that this is also the global optimum.

In general, clustering algorithms can be characterized as: *connectivity-based*, *centroid-based*, *distribution-based*, and *density-based*. The clustering criterion as well as the algorithms are different for each of them.

The core idea of connectivity-based clustering strategies is to regard objects as more related if they are nearby. The clusters are thus formed based on the distance between data points. Algorithms in that category are also known as hierarchical clustering algorithms, as a hierarchy of clusters is provided that are merged when they are close enough. Examples are linkage clustering algorithms.

In centroid-based clustering algorithms, a so-called *centroid* represents a cluster. This centroid does not need to be a member of the dataset. An example for this type of algorithm is *k-Means*. The objective of k-Means is to find k cluster centers and to assign the data points to the nearest centroid in a way so that the squared distances within the cluster are minimized.

Distribution-based clustering algorithms use a clustering model that is based on distribution models. The clusters are created from all data points that most likely belong to the same distribution. An example is the *Gaussian mixture model* where the data is modeled as Gaussian distributions.

The idea behind density-based clustering algorithms is that clusters are seen as areas where the density is higher than in neighbored sparse areas. Data points outside these areas are considered outliers and mostly regarded as noise. An example algorithm is *DBSCAN* (density-based spatial clustering of applications with noise).

3.3 Distance Functions

Many clustering algorithms need a *distance function* for the successful clustering of data points from a dataset. It is also known under the name of *similarity measure*. The assessment of similarities between data points is absolutely necessary for many tasks like cluster analysis and nearest-neighbor classification. Generally, the definition of such distance measures is a very difficult task and requires good knowledge of the data, or of a distance function that has proven to be suitable. Strehl, Ghosh, and Mooney [SGM00] compared different distance measures like

Euclidian, cosine, Pearson correlation, and extended Jaccard with respect to different clustering techniques in the field of web-page clustering to find good manually chosen distance functions.

A alternative solution, which is widely applied in the literature, is to learn a distance function. Often, a weighted linear combination of the single features is used. These weights are then learned from positive and negative examples generated from the training data. The problem is also typically addressed in the context of supervised clustering where we can infer constraints so that data points belonging to the same category have a smaller distance to each other than to other points not belonging to the same category.

Richter [Ric93] analyzes the learning of distance functions for the classification task using a case-based scenario. He formulates the task as the one to find a distance function d such that two data points a, b are in the same class if the distance $d(a, b)$ between both is sufficiently small. A different approach to find suitable distance functions and features is presented by Kira and Rendell [KR92]. The authors use an interactive system allowing users to rate a similarity prediction. The distance function is then enhanced using reinforcement learning. A similar approach relying on the same technique can be found in the publication of Salzberg [Sal91].

Besides these early publications, the learning of distance functions has been widely used in scenarios with different algorithms and techniques spread over different domains. Davis et al. [Dav+07] present an information-theoretic approach for nearest-neighbor classification. Another successful approach to distance function learning, using neural networks, has been presented by Stein and Niggemann [SN01]. Bar-Hillel et al. [Bar+03] and Eick et al. [Eic+05] show that the clustering performance can be massively improved by using Ring Clustering Algorithm (RCA) and k-Means, respectively. Bilenko and Mooney [BM03] present a classification-based approach using a Support Vector Machine for learning similarity measures. In the social media area, there is notable work from Becker, Naaman, and Gravano [BNG10], who compared an ensemble-based approach to an SVM-based approach for a distance function learning, showing that both approaches can be applied successfully. They further show that the SVM-based approach is even more effective than the ensemble-based approach.

Joachims [Joa02] proposes to use a ranking SVM to learn a linear ranking model that can be exploited to rank documents in information retrieval scenarios. Because the task of learning a similarity function from training data is directly related to the task of learning a ranking function in the area of Information Retrieval, in one of our published works we also presented an approach for the social media field using different types of Support Vector Machines (ranking and standard SVMs) [RC11]. This work follows approaches by Joachims [Joa02] and Fakeri-Tabrizi et al. [Fak+10]. Fakeri-Tabrizi et al. [Fak+10] use SVMs for an imbalanced classification problem with image annotation and compare a standard and a ranking SVM, showing that the latter performs better. All approaches show that a ranking SVM can be more stable than a standard SVM. A similar approach using a Perceptron to learn a ranking function has been introduced by Gao et al. [Gao+05].

3.4 Knowledge-based Clustering

We have briefly introduced the concepts of supervised classification and unsupervised clustering. While it is possible to use knowledge that has been acquired in the past in the classification task, it is totally ignored in unsupervised clustering approaches. In recent years, there has been research on introducing another form of clustering which can be summarized under the term *knowledge-based* clustering. In this area, ideas from supervised classification and unsupervised clustering are combined.

The clustering task using labeled and unlabeled data has been dubbed *semi-supervised* [CCM03; B+99; BBM02] clustering. If only labeled data is used for the creation of the model, like in work from Eick, Zeidat, and Zhao [EZZ04], the term *supervised* clustering is used. The process of clustering is significantly different if it is totally unsupervised (without labeled data) or done with a least some kind of supervision (with labeled data).

In the following, we survey clustering approaches where background knowledge is incorporated into the clustering task. The principle idea has already been used in 1996 by Bensaid et al. [Ben+96] for the task of image segmentation. In this clustering setting, unlabeled and labeled data are both used together. In particular, the known class labels or pairwise constraints on several examples are used to aid an otherwise unsupervised clustering of unlabeled data. Kononenko and Kukar [KK07, p. 355] identify two different strategies to incorporate background knowledge of knowledge-supervised clustering: *constraint-based* and *metric-based*.

Approaches falling in the constraint-based category are those where the clustering algorithm itself is changed to get a better and more appropriate clustering. These changes to the algorithm are of different kind, but they all have in common that they change the objective function to satisfy or enforce some constraints. Demiriz, Bennett, and Embrechts [DBE99] modify the algorithm to include a satisfaction of constraints. Similar to that, Wagstaff et al. [Wag+01] add two pairwise must-link constraints (condition that to two data points must be in the cluster) and cannot-link constraints (two data points must not be in the same cluster) between data points to the k-Means algorithm. A comparable pairwise-constrained approach has been considered by Basu, Banerjee, and Mooney [BBM04a], who added an active learning strategy. As a good seeding is important for a lot of clustering algorithms (like k-Means), there are also approaches where the labeled examples are used to initialize the cluster centers (see Basu, Banerjee, and Mooney [BBM02]). Another supervised clustering algorithm has been introduced by Jirayusakul and Auwatanamongkol [JA07], who use a prototype-based algorithm incorporating techniques from unsupervised Growing Neural Gas algorithms. Rendle and Schmidt-Thieme [RS06] incorporate supervision to guide the search for a clustering in the field of record linkage using a Hierarchical Agglomerative Clustering (HAC) approach.

In metric-based approaches, the clustering distortion measure is adapted so that given constraints are satisfied. The clustering algorithm itself is not modified. The main idea is to learn an optimized distance function with the help of labeled data. There are several approaches using different distance metrics. Examples are works from Klein, Kamvar, and Manning [KKM02], who used an Euclidean distance metric which has been trained using a shortest-path algorithm, and Cohn, Caruana, and McCallum [CCM03], who optimized a Kullbach-Leibler divergence using gradient descent. A lot of other researchers have employed approaches with different distance metrics and optimization algorithms (see e. g. Xing et al. [Xin+02] and De Bie, Momma, and Cristianini [DMC03]).

Bilenko, Basu, and Mooney [BBM04c] use a combination of a constraint-based and metric-based clustering approach. They show that the results from their unified approach are better than other approaches where labeled data is used only for one of the techniques individually. The same authors also introduced a framework using Hidden Markov Random Fields where they incorporated supervision. In addition, their framework allows for the optimization of different distance metrics [BBM04b]. Eick, Zeidat, and Zhao [EZZ04] compare different algorithms and their performance with supervision identifying difficulties with many of them.

3.5 Large-scale Processing and Scalability

In many applications datasets are large, which makes processing expensive. This is indeed the case when working with social media data. Following McCallum, Nigam, and Ungar [MNU00], datasets can be large in three different ways: i) the number of elements in the dataset is very large, ii) the feature dimension is high, as every element may have a lot of features, and iii) the number of clusters to be created is high. Social media datasets are to be regarded as large in all of these ways. Therefore, the clustering of large-scale datasets has been identified as an important challenge.

While there are many efficient naive implementations of clustering techniques, they all have different reasons why they cannot be used if a dataset is large in more than one of the above mentioned ways. Most machine learning algorithms are actually only useable in a setting where the dataset is fixed and small. In the following we therefore consider algorithms and methods which are capable of handling large-scale and stream-based data sets.

As we have already discussed before, scalability is crucial. It is an absolutely necessary requirement if the dataset is of large scale and if we are dealing with a stream data setting. We follow Aggarwal [Agg07, p. 44] who provides an overview about methods making a clustering possible. He differentiates two techniques enabling us to cluster large-scale stream data: *task-based* and *data-based*. Techniques falling in the first category address computational challenges modifying or inventing different algorithms, while the ones in the second category refer to a summary or subset selection of the data [GZK05]. In the following we will briefly go into more detail and present some techniques for each category. We dedicate a special section to candidate retrieval which is one form of a sampling strategy, a data-based technique.

3.5.1 Task-based Techniques

Task-based techniques can be summarized as the modification or invention of algorithms with the goal to scale up the clustering process with a possible loss of precision and exactness. Following Aggarwal [Agg07], some prototypical techniques falling in this category are: *approximation algorithms*, and algorithms using *sliding windows* and *output granularity*.

Approximation algorithms are algorithms designed to solve NP-hard problems, as it is unlikely that there are efficient polynomial-time exact algorithms solving these problems [Vaz01]. The results from these algorithms are approximated solutions with error bounds. The aim is to find at least one solution that comes close to the exact solution for problems that are otherwise not solvable within a certain time frame. However, such algorithms still have problems to adapt the resources to the data rate [Agg07], making them not suitable for our problem.

The sliding window techniques are only a range of data in a specific corridor. In this context it means that only the most recent stream is used and analyzed. Dong et al. [Don+03] and Babcock et al. [Bab+02] claim that older data (outside of the window) are summarized versions while the detailed analysis is only done on the most recent data.

The algorithm output granularity has been proposed by Gaber and Yu [GY06] in a sensor data stream setting. They claim that this is a fully resource-aware data analysis approach which is capable to cope with fluctuating and high data rates. The algorithm measures how many resources are available on the system doing a clustering and adapts the number of cluster centers. If the resources are exhausted, a re-clustering is done so that the number of cluster centers is lowered allowing for further clustering.

3.5.2 Data-based Techniques

Data-based techniques process data by summarizing or selecting data points, so that the clustering algorithm itself does not need to handle all data points but only a smaller preselected subset. Common approaches are *sampling*, *load shedding*, *sketching*, and *batch processing* [Agg07].

Sampling is one of the most widely used technique for choosing a subset of data to be processed. Many strategies are discussed by Sudman [Sud76]. In clustering tasks, random sampling approaches are often used, where a random choice is made to decide if a data point is processed or not. In many approaches new data samples are retrieved until convergence is reached. However, in our setting a random sampling strategy is not desirable as the selection of well-chosen samples is crucial for our task. We will discuss a suitable sampling strategy—candidate retrieval—in Section 3.5.3.

Using load shedding techniques, a sequence of data (usually yet unprocessed tuples) is dropped if the system cannot fulfill the demand with its available resources. Babcock, Datar, and Motwani [BDM04] analyze when a controlled load shedding should be performed in a data stream setting, helping to reduce resource consumption while ameliorating results. This raises the question whether load shedding can be done not only randomly but guided by the importance of the content. Tatbul et al. [Tat+03] provide a positive answer to that, showing that this leads to only a minimal degradation in quality. Nevertheless, load shedding is not an option in our scenario even if there are some guarantees. It is undesired to skip data points which are never processed again, as the final result set will be incomplete.

Following Babcock et al. [Bab+02], sketching refers to a synopsis of the data which is held in memory instead of an exact representation. This keeps the computational time to process a data item at a minimum. Such a sketch or synopsis can be seen as a summarization technique [GZK05].

Batch processing refers to a temporary caching of data points together with an aggregation. In such a scenario, data is processed in two passes. In the first pass, data is preprocessed using an aggregation step, producing a preliminary approximation of a clustering for the temporarily cached items. In the second step, the data is processed in a batch mode resulting in the final clustering. Urhan and Franklin [UF00] present XJoin where they use a mitigated version of the technique solving the problem of a too high data point rate. He et al. [He+10] show a successful integration of the batch process in a stream data processing. However, this technique can be used only if the data can be processed in more than on pass.

It is obvious that all presented techniques are likely to produce a result that is approximative; the precision of a clustering will thus not be of 100 % accuracy. While some approaches like load shedding suffer more from this effect, for others the impact is lower (like batch processing where an approximation can be avoided if an intermediate clustering is of 100 % accuracy).

3.5.3 Candidate Retrieval

Many works, for example Guha, Rastogi, and Shim [GRS98], show that one reason for scalability issues is the number of comparisons to make. This has to be addressed using some way to limit the total number of these comparisons.

Candidate retrieval—also known as *blocking* or *candidate generation*—is one possible sampling technique to make standard machine learning approaches scalable. Its origins lie in the task of record linkage, the task to link all data points in a database which share one or more entities and thus should be merged. Nowadays, this technique is not limited to record linkage tasks, but there are a lot of fields where it can be used to speed up retrieval processes. In general, this method can be useful if the number of data points to be compared is too high. Even the best algorithms have a maximum limit of possible comparisons; either this might be due to memory or time constraints, or there will be a point where no more comparisons are computationally feasible. Potentially, and this is the case in many tasks (including our approach), all data items have to be compared to all other data items in the dataset. This means that the number of comparisons grows quadratically with the number of items to be compared.

The idea of candidate retrieval is to restrict the number of comparisons by choosing a subset for which the comparisons are conducted. Rendle and Schmidt-Thieme [RS08] show that using such a step indeed makes approaches more scalable. Using this technique, there are two crucial points which are important: i) the subset must cover the correct matches as good as possible, and ii) it is desirable to throw away as many non-matches as possible. In order to gain performance, the candidates are retrieved at the beginning of each comparison step. This enables us to do the rather computationally expensive comparison only on those data point pairs for which the probability of a positive match is high. The overall goal here is to keep the number of comparisons constant, even though the number of items to be compared grows continuously. Another purpose of candidate retrieval is to improve the data quality itself. The candidate retrieval step is smoothing the dataset by removing unwanted noise and potential false matches.

In order to measure the impact of candidate retrieval Elfeky, Verykios, and Elmagarmid [EVE02] present several tools which allow to measure of the effectiveness and gain of a candidate retrieval step. They provide definitions for the reduction ratio (percent of candidates which have been removed), the pair completeness (how many relevant candidates are included) and overall accuracy (a ratio combining the two others). Having these measures in mind, we follow Michelson and Knoblock [MK06] who frame the goal of candidate retrieval more formally as a high reduction ratio together with a high pair completeness. In general, we desire to reach a high accuracy while keeping the computational complexity low.

Taking a look to the field of record linkage, Baxter, Christen, and Churches [BCC03] present an overview of four different blocking strategies: bigram indexing, canopy clustering with TF-IDF, sorted neighborhood blocking, and the so-called traditional blocking. While the last one is only relevant for record linkage, the others are also useful in our scenario. In work from McCallum, Nigam, and Ungar [MNU00], the number of candidates is limited using so-called *canopies*. These canopies are overlapping subsets originating from an efficient division using a cheap distance measure. Exact distances are then only calculated for the data points laying within a canopy.

As our task is more complex because the data is orders of magnitude larger when using social media data, we need particularly efficient techniques. We thus have to design a special candidate retrieval technique for event identification.

3.5.4 Stream Data

As discussed in the previous sections, large-scale datasets are challenging to process. Stream data is an ordered sequence of data which grows continuously; this type of data is thus to be seen as large-scale. As the data points have a fixed chronological order, a free random access to the data is not possible. The data size is unlimited and not all historical data might be available.

Some authors like Vlachos et al. [Vla+04] and Becker, Naaman, and Gravano [BNG09] cache a data stream to process it ignoring the timely-ordered manner of the data. This can be seen as a shift of the problem to a normal large-scale clustering problem. While this allows for random access to the data and the use of classical data mining algorithms, there is a big disadvantage: the clustering is not available instantly and the latest data points are missing as new data points arrive continuously. Depending on the intended purpose, this can be bothersome and the clustering might be available too late. In addition, these approaches do not deal with the problem of unlimited length of the stream and the memory fitting problem.

Papka and Allan [PA98] and Guha et al. [Guh+00] were one of the first to present clustering algorithms to be applied directly on stream data. While Guha et al. [Guh+00] present an algorithm which uses more than one single pass, Papka and Allan [PA98] strictly adhere to a single pass clustering together with a classification decision for new events at the moment of their appearance. Aggarwal et al. [Agg+04] use a comparable approach where they create a classification system in which an adaptation of the training model to the changes caused by concept-evolution is possible. This is achieved by a regular and automatic retraining of the model using newer data. More recent work from Masud et al. [Mas+11] goes into the same direction. They present an algorithm which addresses the same issues. While Aggarwal et al. [Agg+04] need a delay to identify the concept-evolution, this is working instantly in the more recent work by Masud et al. [Mas+11].

3.6 New Event Detection

The new event detection problem consists of detecting the appearance of a new event in a dataset of event-related data points. This means that data points which do not belong to any event cluster seen so far have to be identified.

One of the first mentions of the new event detection problem in the literature was in the topic detection and tracking (TDT) task [All+98], where it was called *first story detection*. It can be seen as the problem of detecting *outliers* or *anomalies* in the data stream. The aim thus is to detect an abnormal behavior in the data, i. e. that the characteristics of the data point deviates significantly from the regularities of all other data points. Outlier and anomaly detection is important in the field of intrusion detection, fraud detection, robustness analysis of network systems, as well as other data mining applications and has been studied widely. Different approaches have been proposed to solve this problem, among them i) statistical, ii) unsupervised or clustering-based, and iii) supervised or classification-based approaches.

In overview papers from Markou and Singh [MS03a] [MS03b], Chandola, Banerjee, and Kumar [CBK09] and Hodge and Austin [HA04], different approaches and algorithms for novelty detection and outlier/anomaly detection are discussed. The authors show that it is not a well-formulated problem and many researchers tackle it with different approaches depending on the type of data. The authors agree on three types of outlier detection problems (see especially Hodge and Austin [HA04]):

- **Type 1** is the task to determine outliers without prior knowledge of the data. This corresponds to an unsupervised clustering.
- **Type 2** models both normality and the outlier. This can be seen as supervised classification.
- **Type 3** just models normality. This is equal to a semi-supervised clustering.

In the following sections we survey different approaches for outlier detection according to these three main types of approaches.

3.6.1 Statistical Approaches

Initially, the outlier detection problem came up in statistics. The approaches here can be divided into parametric and non-parametric ones. Parametric techniques rely on standard statistical tests such as Grubb's test, the t-test, and χ^2. Already in 1969, Grubbs [Gru69] reported procedures for detecting outlying observations in samples. He also was the first to provide a definition of an outlier which he formulated as follows: "An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs". Aggarwal and Yu [AY01] present an outlier detection method in high-dimensional data based on Grubb's procedures. While Surace, Worden, et al. [S+98] present work an novelty detection using the t-test, Ye and Chen [YC01] propose the usage of χ^2 statistics to check whether a new data point indeed stems from the training distribution or not. Non-parametric approaches are, for example, histogram-based techniques. They do not make strong assumptions about the distribution or density of the data. Examples for histogram-based techniques maintaining a profile of the normal data have been applied in work by Eskin [Esk00] and Fawcett and Provost [FP99].

3.6.2 Unsupervised Approaches

Outliers can also be found as a by-product of an unsupervised clustering algorithm. Here, it is assumed that "normal" data points belong to a cluster while outliers are data points which were not assigned to any of the clusters [GRS98; ZRL96; Est+96]. Usually, if the outliers are not assigned to any cluster, they are considered as noise. However, in many tasks outliers are valuable information, as it might indicate a change of concept such as the introduction of a new event.

Many clustering-based approaches attempt to discover low-density regions in the data space, assuming that outliers belong to a cluster with a density below a certain threshold. Such approaches are typically called density-based. Work presented in this area is e. g. by Aggarwal and Yu [AY08], who use a probability of presence of a uncertain data point in a sparse region to determine outliers. A similar approach using a *local outlier factor* has been introduced by Breunig et al. [Bre+00]. Makkonen [Mak03] uses hierarchies to track event evolutions and if the density is too low, the detection of a new event is assumed. Frequent item set mining has also been applied to detect normal patterns in training data, regarding all those data points as outliers that do not fit the frequent item set patterns [BLC02].

Most of the cited works have in common that all outliers can be detected only after the whole dataset has been seen; these are thus not applicable in a stream data setting. But, other authors have proposed online methods and applied them to sequence data [BY01; Bud+06]. As shown by Allan et al. [All+98] for the topic detection and tracking task, new event detection is particularly interesting in the scope of temporal stream data. We agree with Allan et al. that the aim is to detect the first appearance of a new event in a data stream. In this context, the arrival of a new event can be denoted as *concept evolution*, as the concept of the new data point is different from the ones in the past. Commonly, the techniques used here work within single-pass settings.

Other approaches rely on distance-based criteria [KN98; RRS00; VG04]. Papka and Allan [PA98] propose to use a threshold model that incorporates time distances in order to decide on a new event. Knorr and Ng [KN98] define an outlier as follows: "[a] point p in a data set is an outlier with respect to the parameters k and λ, if no more than k points in the data set are at a distance λ or less from p". That this approach requires λ to be chosen manually can be problematic in the sense that if the value is chosen to large or small, too few or too many data points are considered as an outlier. Newer approaches like the one from Kriegel, Schubert, and Zimek [KSZ08] use a different strategy to overcome this problem, using an angle-based approach with the side benefit of a better scalability and performance. Others propose different solutions, for example, Aggarwal and Yu [AY05] use a lower-dimensional projection, while Gao et al. [Gao+10] propose a generative mixture model. Masud et al. [Mas+10] investigated data streams in a dynamic feature space, introducing DXMiner which uses a sliding window.

3.6.3 Supervised Approaches

Supervised or classification-based approaches use discriminative techniques to separate low-density from high-density regions in the dataset. The approaches in this area have in common that integrating a new event detection mechanism into traditional classifiers ends up in a supervised clustering setting. Examples in the field of news text classification can be widely found in the literature. For example, there are approaches that apply SVMs, Neural Networks, and Bayesian Networks in this context [CBK09].

Ratsch et al. [Rat+02] consider the problem of outlier detection as a one-class learning problem. They use radial basis function (RBF) kernels to define complex regions that contain the training data instances. For each test data instance, it is determined if the instance falls into this region, regarding it as an outlier if it does not. Work by Davy et al. [Dav+06] goes into the same direction; they show a successful application of a Support Vector Machine to the problem of detecting abnormal events. In contrast to Ratsch et al., they do not rely on only one class.

Other approaches for detecting novelties use Neural Networks. In order to determine whether a data point is an outlier, they test whether the data point is accepted by the network as an input or not. If it is not accepted, it is regarded as an outlier [MS03b]. An approach called *Long-term Depressant SOM*[1] (LTD) was presented by Theofilou, Steuber, and Schutter [TSS03]. Here, the weight vectors of the cells in the map become more and more dissimilar to the seen training instance over time. The LTD-like SOM works differently from the classic Kohonen network. Instead of a movement of the weight vectors closer to the input distribution, the weight moves away from it. Newly arriving data points which are close to these weight vectors are regarded as outliers in the LTD-like SOM.

Kumaran and Allan [KA04] as well as Zhang, Zi, and Wu [ZZW07] tackle the problem of new event detection in the same context as proposed in the topic detection and tracking program [All+98]. Kumaran and Allan experiment with the representation of the data in the vector-space model introducing named entities. The approach from Zhang, Zi, and Wu is also based

[1]Self-organizing map

on named entities and similarly uses indexing-trees for a faster retrieval. This is the first approach focusing on scalability.

3.7 Event Identification and Detection

In recent years, a lot of research has been conducted in the field of event identification and detection using the techniques discussed in the previous sections. Event identification and detection comprises several procedures with different objectives. We will discuss the different objectives first:

(1) Event Identification: Event identification is the process of identifying if an event occurs. We use this term to denote the task of distinguishing documents representing an event from those that do not.

(2) Event Category Detection: The aim of event category detection is to categorize documents into general event categories, like "soccer match" or "festival", which are part of a taxonomy.

(3) Event Detection: In event detection, documents identified to represent some event are to be classified into specific events. The difference between event category detection and event detection is that in the latter context the event instances are unique and specific like "soccer match between Russia and Japan in March 2010" or "Great Ocean Festival 2013".

In the literature, the terms are often used interchangeably. To our knowledge, the term event identification has also been used to describe the objective (1) and (3). In the following we use the above defined notions to clearly distinguish between the approaches. The approach presented in this dissertation aims to detect specific events and therefore clearly addresses the third objective—event detection.

Earlier, event identification and detection has been done by hand or with very simple algorithms. A system by Sarvas et al. [Sar+04] called MobShare allowed a manual assignment of image folders from friends to one event, but the creation and identification of the events as well as the merging process were completely manual. Many (semi-)automatic approaches use temporal information for event detection, e. g. for clustering a digital photo collection [Coo+05; Gra+02]. The effort involved in these approaches is very different. One uses simple cuts through a predefined time threshold, e. g. Apple iPhoto, a popular photo managing application, allows to automatically cut photo collection into slices of 2, 8, 24 hours, or a week. The other uses different kind of unsupervised and supervised clustering approaches (see Cooper et al. [Coo+05]). It has been shown that the sole use of temporal information works quite well for single-user applications. According to Cooper et al. [Coo+05], the F-measure for a Bayes information criterion-based boundary selection is very high using two different datasets. They also show that the same applies for a simple time threshold (being the best in one dataset). Arguably, the use of only the temporal feature would not work sufficiently well using social media content shared by a lot of people. Nevertheless, these findings are a basis for our multi-pass clustering approach (see Part III).

Social media content has received a lot of attention in the past years. Since many documents in this area represent some type of event, it is especially interesting to do event identification and detection with this type of data. Previous work has already been shown to bring order to social media data. Since 2011, as part of the MediaEval benchmark, an initiative dedicated to evaluating new algorithms for multimedia access and retrieval, there is a challenge called *Social Event Detection* (SED) requiring participants to discover event-related multimedia. Over the years, the challenges dealt with different kinds of event identification and detection tasks [Pet+14]. While in 2011 and 2012 the focus was mainly on event category detection, it includes different tasks for all three different event identification and detection objectives since 2013.

In the field of event identification, several attempts have been made to decide whether a social media document is associated with an event or not. Research in this area is also known under the name *burst detection* [HCL07]. Chen and Roy [CR09], for example, present an approach which is able to distinguish between aperiodic and periodic events. In that sense, their work is related to the one by Rattenbury, Good, and Naaman [RGN07], who present a method called "Scale-structure Identification (SSI)". They are able to apply the method to identify tags representing events as well as places. In more recent work, Rattenbury and Naaman [RN09] present an attempt to learn a classifier that distinguishes not only events and places but also social media documents from Flickr representing an event and documents that do not. Firan et al. [Fir+10] use different Naive Bayes classifiers to decide whether an image is associated with an event or not. Others proposed to use wavelet-based detection methods to identify events on Twitter [WL11]. The results of all these approaches show that improvements are still necessary, as the detection either works only for small domains or there is a large portion of misclassified documents. In the SED 2013 challenge, there were better approaches using mainly Support Vector Machines as classifier. One approach reached a decent result by employing a very rich set of textual features together with a set of ontological features [Ngu+13].

A first approach to cluster documents from Flickr into event categories has been proposed by Firan et al. [Fir+10], who use a Naive Bayes Multinomial classifier assigning images to Wikipedia and WordNet categories. Barnard and Forsyth [BF01] use a generative hierarchical model integrating semantic and visual information to create an image database browsing system. In the second MediaEval SED 2013 challenge, participants were asked to cluster a set of images into eight event categories. The participants adopted a direct classification procedure [BI13; Ngu+13; SN13]. With respect to the results, all approaches still have severe problems to determine the correct categories; the best only reached an F-measure of 44.9 %.

One of the first to tackle the problem of event detection in social media were Becker, Naaman, and Gravano [BNG10], who presented an incremental clustering algorithm that clusters social media documents into a growing set of events. The clustering algorithm used is similar to the one used for detecting events in text documents streams (see Allan et al. [All+98]). In this approach, all documents have to be compared to each other before the assignment to the final category can be made. This causes scalability issues as these comparisons grow linearly with the data. There has also been other work on detecting and indexing events in social media streams like the one by Chen and Roy [CR09], who consider an approach for detecting events exploiting the temporal and spatial distribution of keywords and tags.

More recently, we have proposed an event detection task on the image dataset described in this dissertation to be used in the Social Event Detection task of MediaEval 2013 (see Reuter et al. [Reu+13]). In addition, we provided an evaluation tool to make all results comparable. There was a total of 11 participants in that task. Therefore, we are able to directly compare our approach presented in this thesis to the approaches of the MediaEval participants. As the task did not require to treat the data in a stream scenario, it was possible to use the dataset in a fixed setting allowing a back and forth search with no limitation in the number of passes. In the following we will briefly give an overview of the different approaches that were applied.

In general, it was possible to use two kinds of data for each document: *metadata* and the visual image itself. The metadata included the capture time, upload time, geographic information, tags, title, description, and information about the person (distinct identifier) who uploaded the image and other data. It was possible to use any feature combination, and not all features had to be used. While all approaches made use of the metadata, only some approaches additionally used the image data, which they represented by SIFT or SURF features. All these approaches showed that the use of visual information does either not ameliorate the overall quality of clustering or, worse, even lowers the overall performance [Raf+13; BI13; Sch+13]. Some approaches [ZZD13; WS13; SN13] heavily preprocessed the textual metadata by applying stop word removal, stemming and/or non-ASCII character removal. Nevertheless, the results were not as good as the ones from others who did not apply any text manipulating techniques. Others rounded the timestamps to higher time spans, up to whole days. While the impact of a smaller rounding is not measurable, the rounding to a full day may cause problems as the results were a lot worse [GGC13].

All researchers were faced with the scalability issues caused by the number of data points in the dataset. Some did not deal with the problem at all, resulting in a slow and not very scalable approach. Others tackled the problem by using very different techniques. A few made use of a candidate retrieval step, which has been proposed in earlier work by us [RC12] which will be explained in more detail in Section 5.3. While some, like Schinas et al. [Sch+13] and Manchon-Vinzuete and Giro-i-Nieto [MG13], directly adopt the candidate retrieval method we have proposed, others used it as an inspiration for a variant using only the time factor [Ngu+13]. Another idea proposed was to use a search engine like Lucene or Sphinx to create the candidates [Sam+13]. A different solution to tackle the scalability problem was to split the dataset into batches in order to make processing possible for a computationally time expensive algorithm [WS13].

Many of the approaches use algorithms which cannot be used in a stream setting as they are opting for a sequence of several clustering operations which cannot applied to data streams [ZZD13; Raf+13; Ngu+13; MG13]. It is common to cluster by time and location first (e. g. Zeppelzauer, Zaharieva, and Del Fabro [ZZD13] and Gupta, Gautam, and Chandramouli [GGC13]) and then iteratively incorporate other features in one or more passes. There are also approaches which initially consider a per time and user cluster, subsequently adding another merge algorithm [MG13; Ngu+13]. The results in SED 2013 show that multi-pass algorithms seeing the whole data set (all data points) at once have a higher performance than algorithms with limited passes for a stream setting. For example, Samangooei et al. [Sam+13] present a spectral clustering algorithm with the density-based structural clustering algorithm for networks (DBSCAN) with neighborhood selection clustering. They preprocess data in a sparse affinity matrix, achieving a very good performance (reaching an F-measure of 94.6 %).The approach by Wistuba and Schmidt-Thieme [WS13] uses a Quality Threshold clustering. Instead of using a document-document based representation, they were inspired by earlier work of us, using a document-centroid based representation. An approach applicable in a stream setting was presented by Manchon-Vinzuete and Giro-i-Nieto [MG13]. They propose a two-pass approach using Photo-TOC [PCF03] as a first step to cluster pictures by time for every user, and then use a merge algorithm to produce the final clustering (F-measure: 88.3 %). Later, in Section 9.3, we provide a direct comparison of performance with the most important approaches mentioned in this section with our approach when presenting our results.

CHAPTER 4

Event Clustering Dataset

In order to create, test, and validate the social media clustering approach of this work, it is necessary to have a suitable dataset. There are already many datasets published which help researchers with developments in the area of classification and clustering to test their approaches. Examples are the datasets from Chapelle, Schölkopf, Zien, et al. [C+06] who proposed three artificial and three real datasets for testing and validation of semi-supervised clustering approaches. There are also several datasets published based on the Reuters news corporation collection [San94]. One example for such a text collection is the Reuters-21578 dataset [Lew97]. There are at least two datasets in the social media field which we are aware of. Over et al. [Ove+12] presented a dataset comprising videos as part of TRECVID MED. Another dataset comprising images has been proposed by Papadopoulos et al. [Pap+13a] in the MediaEval SED challenge. However, many of these datasets (including the ones mentioned above) have several drawbacks that limit their use for our approach: either they are totally artificial, not related to social media, or the number of documents/data points included is small. For example, the dataset from Papadopoulos et al. [Pap+13a] only contains a little more than 100,000 images and they provide a classification only for a fraction of these. Furthermore, the classification is based on event categories and is thus not suitable to evaluate an approach for distinct event detection. We thus decided to create a new dataset. As the aim is to develop an application for a real-world instead of an artificial application, the dataset should be non-toy and reflect real-world data. Therefore, in the following we propose such a dataset which has already been released in 2014 under the name “ReSEED – Social Event dEtection Dataset” [Reu+14].

Our dataset consists of pictures from the online photo sharing community platform Flickr[1] which is operated by Yahoo Inc. All the pictures are assigned to individual and distinct social events. These social events are of large variety and include concerts, festivals, sport events like soccer matches, protest marches, debates, expositions, and others.

In addition to the pictures itself, Flickr provides so-called *metadata* for each picture. In our case, we are talking about *descriptive metadata*. As its name already implies this data describes

[1]`http://www.flickr.com`—last accessed 2014-09-19

the data file itself. More precisely, in this context metadata is textual data like a timestamp, keywords, title, as well as a textual description of the image content.

The whole dataset including all pictures, together with their associated metadata, was downloaded from Flickr using their official API[2] which provides a convenient way to access the official data. Furthermore, we made sure that all pictures have been published under a Creative Commons license. This type of license granted by the image uploaders allows a free distribution of the pictures and the metadata even without the usage of Flickr.

In what follows we give information on how exactly the data was obtained. The creation process of the event clusters themselves is then described in section 4.2. After that, we proceed to describe the dataset in more detail and provide necessary figures and statistics. We conclude with a description of the dataset format.

4.1 Creation and Collection of the Dataset

In order to collect the data material we relied on the official API from Flickr. Altogether, the retrieval of the images and their accompanying metadata has been performed in four subsequent steps:

1. All metadata available for every photo in the dataset was fetched using the function `flickr.photo.search` of the Flickr API.
2. All available information about the photographer and uploader of the photos has been fetched using the `flickr.people.getInfo` function.
3. The photos themselves were downloaded via standard HTTP requests using their addresses, which were fetched in the first step.
4. Additional information about the events has been fetched from the online event calendars Upcoming[3] and last.fm[4].

For the creation of the gold standard we have used an essential key function from Flickr. Every user of the Flickr services has the possibility to assign so-called *tags* to the pictures which were previously uploaded. These tags are meaningful keywords describing each image. Usually, the tags contain human-readable textual information. This might be a single word, a short slogan (multi-word), or just some character combination with up to 255 characters. In 2007, a special type of tags which are meant to be consumed by machines was newly introduced by Flickr. These special tags are called *machine tags*. Sometimes they are also known under the name *triple tags*. Their main difference to the normal and user-centric keywords is that machine tags are built up using a fixed schema. This schema uses a namespace, predicate, as well as a value. Such a machine tag is denoted in the following form:

[2]`http://www.flickr.com/services/api`—last accessed 2014-09-19
[3]`http://upcoming.yahoo.com`—last accessed 2013-03-10
[4]`http://www.last.fm`—last accessed 2014-10-03

```
namespace:predicate=value
```

A namespace denotes a facet or a class which the tag belongs to. The predicate is used to name the property for a namespace and the value is some value suitable for the property. How this can be used for the creation of the dataset will be explained in more detail in Section 4.2.

4.1.1 Fetching of Metadata

Our first objective was to fetch the metadata for suitable images. We thus exploited an API function as the first step which allows to search for images meeting certain criteria. A suitable function was `flickr.photo.search`. The result of this API call is a set of textual metadata about the photos. It also includes the address from where the image itself can be downloaded. Our intent was to download all photos uploaded to Flickr between the 1st of January 2006 and the 31st of December 2012 fulfilling certain criteria: a) the photo has a special machine tag assigned and the namespace of that tag is either `upcoming:event=` or `lastfm:event=` and b) the license of the image is one of the following Creative Commons licenses: *Attribution alone* (CC BY), *Attribution and ShareAlike* (CC BY-SA), *Attribution and Noncommercial* (CC BY-NC), or *Attribution, Noncommercial, and ShareAlike* (CC BY-NC-SA). Other pictures are not included in the final dataset. This applies especially for images under other licenses like the Creative Commons ones not allowing derivatives (CC BY-ND and CC BY-NC-ND) which are excluded. Limiting the selection of images to the ones under a Creative Commons license has several advantages: a) we are allowed to freely distribute the dataset, b) the licenses are well-known and c) their effectiveness has been already proven by some courts in different countries.

Using Flickr's API there is a maximum of 500 results per call. Even though the number of retrieved picture metadata is not officially limited using the `flickr.photos.search` function, we were faced with the problem that, after a certain amount of picture data (about 5,000 to 10,000) retrieved, the API repeatedly returns the last 500 photos as duplicates. We therefore changed the strategy to calling the API function to retrieve the pictures for one single day only, successfully circumventing the faulty behavior of the API. The final API call thus uses three constraints: a) the machine tag is in one of the following namespaces: `upcoming:event=` and `lastfm:event=`, b) the photo has been taken on the day specified in the query, and c) the pictures was published under a Creative Commons license allowing free redistribution. The exact approach for the whole retrieval process is shown in algorithm 4.1

4.1.2 Fetching of Uploader Information

In step 2 we fetch information about the persons who uploaded the pictures to Flickr. Therefore, we incorporate the API once again. This step is necessary as the Creative Commons licenses used for the pictures request that the original author has to be credited. We thus employ the following function of the Flickr API to obtain the information:

```
flickr.people.getInfo
```

Algorithm 4.1: Metadata retrieval from Flickr

```
Input  : Two namespaces: upcoming:event= and lastfm:event=
Output : Retrieved pictures and metadata matching query
Day ← '2006-01-01';
while Day ⩽ '2012-12-31' do
    foreach Namespace do
        PageNumber ← flickr.search.photoSearch (Day, Namespace);
        foreach PageNumber do
            CurrentPicture ← flickr.search.photoSearch (Day, Namespace, PageNumber);
            foreach CurrentPicture do
                if CurrentPicture is new then
                    Store metadata to local database;
                else
                    Discard picture information;
                end
            end
        end
    end
    Day ← Day + 1;
end
```

The uploader information is requested for each picture entry in our local database using the unique user ID which has been provided by Flickr for each picture. As a result we get the following information about the persons:

- Unique ID on Flickr
- Username chosen by the user
- The real name (if the user has provided that name)
- A link to the user profile on Flickr
- The location where the user is situated (if provided by the user)

At the end of this step, we have stored the uploader information together with the metadata for 450,000 pictures.

4.1.3 Fetching of Picture Files

Using the addresses obtained in the first step and stored in our local database, all pictures are downloaded via the GET function using a standard HTTP connection. During the download phase we saw that around 15,000 of the pictures are not available for download for several reasons. This was the case for some pictures due to unavailability of the Flickr servers. However, the largest portion was unavailable because of a removal by the uploader or copyright holder or changed privacy settings preventing a download. This was the case as the API does not necessarily deliver the latest information. Finally, the whole download process eventually led to a total of 437,370 pictures downloaded.

4.2 Labeling of the Data – Creation of the Gold Standard

In this section we describe how exactly the gold standard was created. A manual labeling of data for such a high amount of pictures by hand is a very time-consuming and expensive undertaking and we came to the conclusion that this is not feasible. We therefore exploited existing manually created and readily available social events which were already defined collaboratively by a community of users. In fact, there exist websites on the Web that host social event calendars.

4.2.1 Usage of Social Event Calendars for Data Labeling

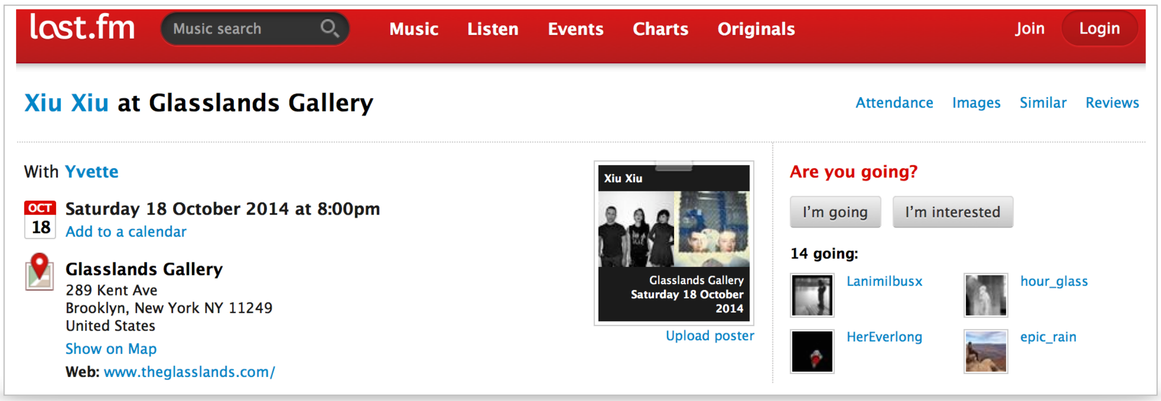
A so-called *social event calendar* is defined as a repository of social events which can be searched and browsed by events. In such a social event calendar, humans create an entry for each distinct event. Services offering this functionality are often managed professionally. The benefit here is that only social events that have been validated by the community are added. Therefore, all included events extracted in this way have a rather small level of noise. Furthermore, it is notable that most of these sites provide more information about the social event than the plain event name. It is very usual that the entries contain more detailed information about the event type itself. This includes the time and date when the social event takes place, the location, a detailed description, and a lot more. A full list of available information is provided in Table 4.1.

For the creation of the dataset presented here, we focused on two services providing a social event calendar: i) Upcoming[5] and ii) last.fm[6]. The first service mentioned here, Upcoming, formerly was a public Internet page operated by Yahoo Inc. It provided event calendar information since 2003. The site has been taken down unexpectedly on April 30, 2013 and the information from the site is no longer available to the public. Fortunately, we still are in possession of a data copy in a local database. Last.fm, the second service claiming to be the world's largest online music catalogue, provides an event calendar since 2007. While the calendar from last.fm targets events related to music, Upcoming provided also entries for all other varieties of events; this included sport events, conferences, protests, etc. The information of the events available from both services is comparable.

An example for a social event presented on last.fm is shown in Figure 4.1.

Both services provide an API enabling us to easily download the event information together with all details available. There is usually more information available using the API than what

[5]`http://upcoming.yahoo.com`—last accessed 2013-03-10
[6]`http://www.last.fm`—last accessed 2014-10-03



Xiu Xiu at Glasslands Gallery

With Yvette

OCT 18 Saturday 18 October 2014 at 8:00pm

Glasslands Gallery
289 Kent Ave
Brooklyn, New York NY 11249
United States

Web: www.theglasslands.com/

Xiu Xiu
Glasslands Gallery
Saturday 18 October 2014

Are you going?

14 going:
Lanimilbusx
hour_glass
HerEverlong
epic_rain

Figure 4.1.: Example from last.fm

is shown in Figure 4.1. For the creation of the labeled data, the most interesting and valuable information is the unique event identifier (Event-ID) for each individual event entry; this is a simple integer value but is the key used to uniquely identify an event. It thus can be used as the basis for creating a gold standard (as originally proposed by Becker, Naaman, and Gravano [BNG10]).

4.2.2 Fetching of Event Information from Upcoming and Last.fm

In addition to the data from Flickr, we obtained information about the events that these pictures depict using the services of Upcoming and last.fm[7]. Both services offered an API which can be employed for the task. Therefore, we used the `Event.getInfo` function form last.fm's API and the unfortunately no more existing equivalent from Upcoming to fetch several information about the events. The features available from each service can be seen in table 4.1.

At a glance, we obtained detailed information for 5,236 out of 5,384 events from Upcoming and 15,043 out of 16,334 events from last.fm.

4.2.3 Labeling Process

The introduction of machine tags in Flickr enabled users to associate keywords with their photos. Machine tags are automatically recognized as a special tag by Flickr when they are entered by the user using the special schema for the triple tag `namespace:predicate=value`. It is possible to use these machine tags in several ways. On the one hand, they can be used for an easier search. On the other hand, machine tags can be used to link data, e. g. pictures or texts, across sites. This is where the unique identifiers from Upcoming and last.fm come into place. The users of both sites were attending several of these social events. At the event they were taking pictures with their cameras and mobile devices which they afterwards upload to their Flickr

[7] `http://www.lastfm.de/api`—last accessed 2014-10-02

Table 4.1.: Available information from two social event calendars, last.fm and Upcoming

Information	last.fm	Upcoming
Unique ID	✓	✓
Event title	✓	✓
Event tags and keywords		✓
Event description		✓
Start time	✓	✓
End time		✓
Number of attendees	✓	
Name of venue	✓	✓
Venue address	✓	✓
URL to venue	✓	✓
Exact geographic location	✓	✓

photo album. Both social event calendar sites began to ask their users to add a well-formed machine tag to the uploaded pictures on Flickr so that other people could use this unique ID to find the pictures or to use the pictures on some other service. The machine tags for these pictures are in the following form: `upcoming:event=#eventid` or `lastfm:event=#eventid`. This provides important information about the pictures for us as we then know that the picture with such a tag belongs to a social event in the database of the social event calendar provider. Subsequently, we aim to obtain the corresponding unique Event-ID for this event. Therefore, we can make the assumption that all pictures which were previously marked with the same Event-ID in the machine tag belong to the same event [RC12] (see also Becker, Naaman, and Gravano [BNG10] for a previous approach exploiting this same principle that inspired this approach).

Being in possession of that information, we went through each picture extracting the associated machine tags. All machine tags being outside of the namespace we are looking for ("upcoming" and "lastfm") were discarded. Afterwards, the *value* of each triple tag for the machine tags `upcoming:event=`*value* and `lastfm:event=`*value* was extracted. The resulting values were then stored together with each Flickr picture ID in our local database. We end up with three different types of relations for the pictures:

1. Pictures having only a relation to the Upcoming event calendar
2. Pictures having only a relation to the last.fm event calendar
3. Pictures having a relation to both services

If an event was listed on both event calendar systems, the event in question was merged to one distinct event. The information about these correspondence was used from pictures having a relation to both services. In the case that a picture only had a relation to one event calendar provider, the pictures were associated with the newly merged event. After that step, the final number of events for the whole dataset is 21,169.

4.3 Dataset Statistics

The dataset as a whole contains 437,370 pictures from the Flickr photo community site, which are available under a Creative Commons license allowing derivatives and distribution with an upload time to the Flickr site between January 2006 and December 2012. These pictures were assigned to 21,169 events in total. All events in the dataset are heterogeneous with respect to type and length. This means, it includes festivals which have a duration of several days as well as soccer matches with only a few hours of duration.

The dataset includes the pictures themselves together with metadata about each image. The data has not been post-processed in any fashion. There are some features in the metadata like upload time and information about the person who uploaded the pictures which are available for every picture without exception. However, as it is a real-world dataset, most of the associated features are only available for a subset of the pictures. In particular, this applies to the following metadata and information about the images that has been fetched directly from Flickr using their API:

- *Unique ID*: This is a unique identifier for the picture assigned to it by Flickr after its upload.
- *Uploader Information*: Information about the person who uploaded the picture to Flickr. This includes a short ID as well as the real name, location, and other data about the uploader.
- *URL*: Location on the Flickr servers where the picture can be downloaded from.
- *Capture Timestamp*: This denotes the date and time when the picture has been taken with the camera device.
- *Upload Timestamp*: Timestamp (date and time) when the user has uploaded the picture to the Flickr community platform.
- *Geographic Location*: Geographic location specified by longitude and latitude.
- *Tags*: All tags and keywords which are assigned to the photo. This also includes the special machine tags.
- *Title*: The title of the picture as chosen by the uploader.
- *Description*: A description of the photo. This is a longer free text provided by the uploader.
- *Views*: The number of persons who have viewed the picture from the upload date till 2013-03-30.
- *License Type*: Type of Creative Commons license (as indicated).

Table 4.2.: Availability of features

Feature	Availability
Upload time	100.0%
Capture time	98.3%
Geographic Information	45.9%
Tags	95.6%
Title	97.6%
Description	37.8%
Uploader Information	100.0%

4.3.1 Data Quality

In spite of the fact that many pictures in the dataset include EXIF data directly from the camera, this data is not directly used in the creation process of the metadata. Nevertheless, the EXIF information was used by Flickr previously to create the metadata—e. g. the upload time is taken from the EXIF information. Even though the EXIF information was not used, it is still associated with the images.

If no useful data is associated with the picture, i. e. if there is no information about when the image has been captured, Flickr uses the upload timestamp also for the capture timestamp. In the download of the metadata via the Flickr API, the capture timestamp is thus unreliable. In order to ensure a good quality of the data, the capture time has been removed from the capture timestamp field if it equaled the upload time.

The metadata about time information, geographic location, and tag information is usually of good quality. However, as it is a real-world dataset this information may be incorrect and there is no guarantee that the user took care of the correctness, e. g. if the time and date configured in the user's camera were wrong, the data will contain false time information. After a thorough overlook of the dataset, we discovered that the title of the pictures very often contains the filename; this is problematic in the sense that it is named by the camera, e. g. "DSCFxxxx", "IMGxxxx", etc. This may mislead a learning algorithm. Also, the description field is used by some uploaders to advertise themselves, to give additional copyright notes, and not to give a description of the shown image.

As not all photo camera devices are equipped with an internal GPS and people do not necessarily tag the geographic location afterwards, only about a little less than half of the dataset has a geographic position assigned. The same applies for other features as only the uploader information and time is a must on Flickr. Exact statistics for each metadata feature are given in Table 4.2.

Table 4.3.: Use of license

License	Fraction
CC BY	20.1%
CC BY-SA	15.4%
CC BY-NC	17.2%
CC BY-NC-SA	47.3%

Table 4.4.: Distribution per year

Year	Fraction
2006	4.6%
2007	21,0%
2008	25,7%
2009	21,4%
2010	13,5%
2011	8,8%
2012	5,0%

4.3.2 License Constraints

As we already have mentioned above, all pictures are licensed under a Creative Commons license. The exact figures of the distribution of the used licenses are given in Table 4.3. We only included pictures in the dataset which allow for free distribution, remixing, and tweaking. Therefore, we only used the subtypes of Creative Commons that allow to use the pictures along the lines mentioned above as long as the owner is credited for his photograph; in particular these licenses are:

- Attribution (CC BY)
- Attribution Share Alike (CC BY-SA)
- Attribution Non-Commercial (CC BY-NC)
- Attribution Non-Commercial Share Alike (CC BY-NC-SA)

The two licenses Attribution No Derivatives (CC BY-ND) and Attribution Non-Commercial No Derivatives (CC BY-NC-ND) were not used to ensure that the pictures and metadata can be changed for research purposes. The same applies for all other license types which are available on Flickr.

4.3.3 Data Point Distribution

The relative number of pictures per year is shown in Table 4.4. It is wroth noting that the distribution over time is not constant. A reason for this is the decreased usage of the online event calendars as the main communication and information sharing about social events has been translocated to other social community sites.

The distribution of the pictures per event is also not uniform. In Figure 4.2 this distribution is depicted for the events with up to 40 pictures per event, clearly showing that the size of the events varies a lot. While 3,598 events include only one single image and 1,799 event include just two pictures, there is only a very small number of events which comprise of over 40 pictures.

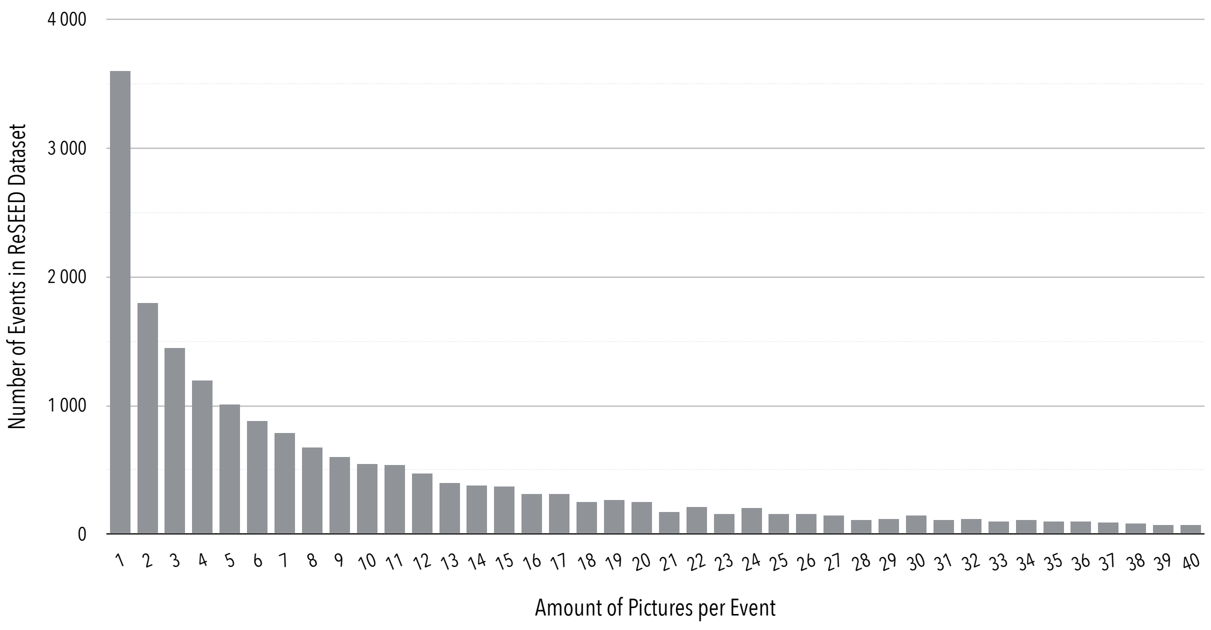


Figure 4.2.: Distribution of pictures per event

This is very challenging as not only the number of clusters but also the cluster size is unknown beforehand.

The 437,370 pictures were uploaded by 4,926 Flickr users, thus corresponding roughly to 90 images uploaded per person. We also observe that 2,418 users uploaded only one single event.

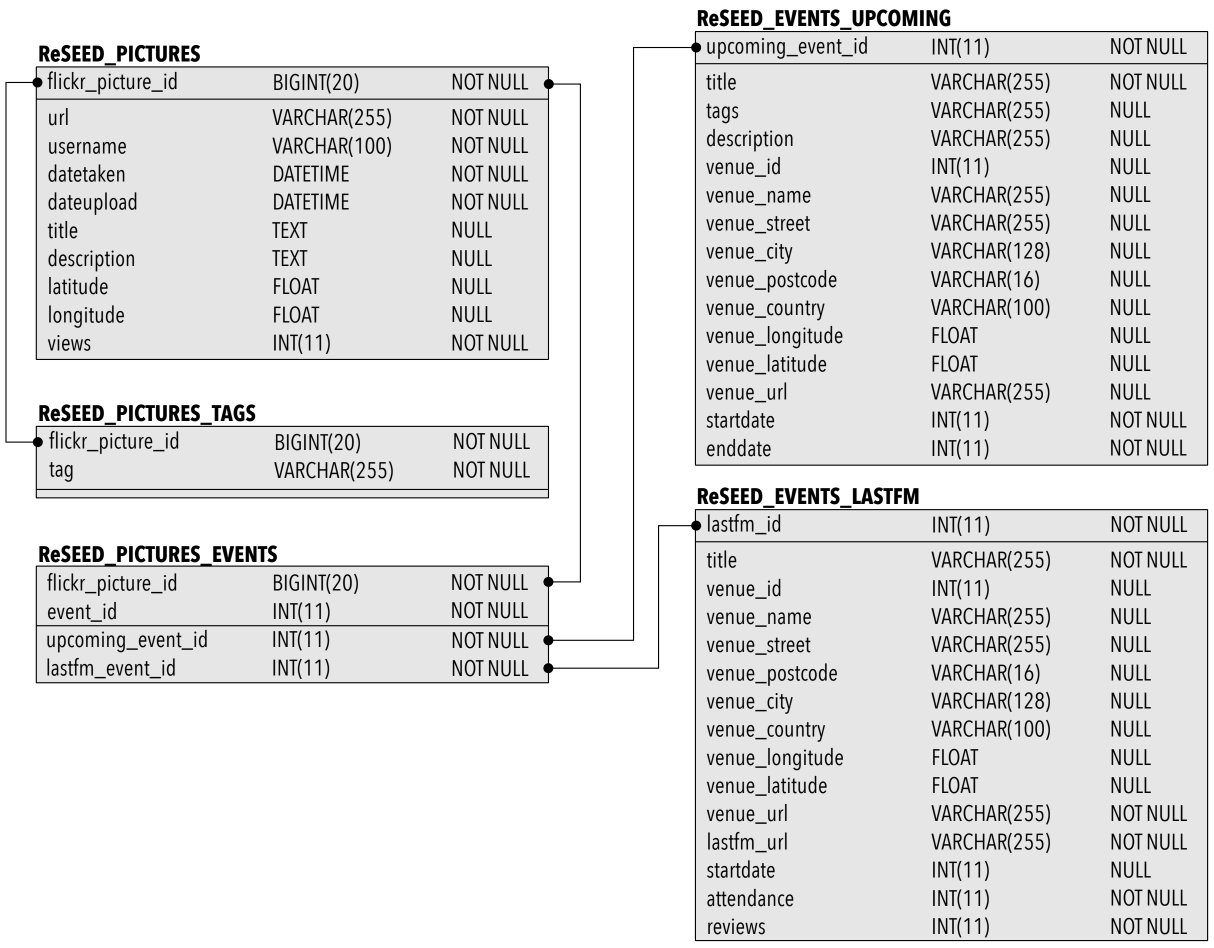
The total number of tags assigned to the images is 3,444,612. There are 91,219 unique tags. About 32.7 % of all tags only have one single occurrence in the whole dataset. This is not surprising as the number of single picture events is also high. This leads to the assumption that there is a high correlation between both.

4.3.4 Dataset Representation Format and Schema

The pictures included in the dataset are stored using the JPEG image format with a maximal size of 1024 pixels for the longer and 768 pixels for the shorter side. The files were directly downloaded from Flickr. The original EXIF information and other metadata incorporated in the files by the uploader or the uploader's camera are still intact and can be used in addition to the already extracted metadata.

The extracted textual metadata of the pictures is stored using a database structure. The schema with the corresponding database tables for this data is shown in Figure 4.3.

The table `ReSEED_PICTURES` is the main and most important table. All relevant picture metadata of the dataset is listed in this table using the unique Flickr ID of each picture as the primary



ReSEED_PICTURES

flickr_picture_id	BIGINT(20)	NOT NULL
url	VARCHAR(255)	NOT NULL
username	VARCHAR(100)	NOT NULL
datetaken	DATETIME	NOT NULL
dateupload	DATETIME	NOT NULL
title	TEXT	NULL
description	TEXT	NULL
latitude	FLOAT	NULL
longitude	FLOAT	NULL
views	INT(11)	NOT NULL

ReSEED_PICTURES_TAGS

flickr_picture_id	BIGINT(20)	NOT NULL
tag	VARCHAR(255)	NOT NULL

ReSEED_PICTURES_EVENTS

flickr_picture_id	BIGINT(20)	NOT NULL
event_id	INT(11)	NOT NULL
upcoming_event_id	INT(11)	NOT NULL
lastfm_event_id	INT(11)	NOT NULL

ReSEED_EVENTS_UPCOMING

upcoming_event_id	INT(11)	NOT NULL
title	VARCHAR(255)	NOT NULL
tags	VARCHAR(255)	NULL
description	VARCHAR(255)	NULL
venue_id	INT(11)	NULL
venue_name	VARCHAR(255)	NULL
venue_street	VARCHAR(255)	NULL
venue_city	VARCHAR(128)	NULL
venue_postcode	VARCHAR(16)	NULL
venue_country	VARCHAR(100)	NULL
venue_longitude	FLOAT	NULL
venue_latitude	FLOAT	NULL
venue_url	VARCHAR(255)	NULL
startdate	INT(11)	NOT NULL
enddate	INT(11)	NOT NULL

ReSEED_EVENTS_LASTFM

lastfm_id	INT(11)	NOT NULL
title	VARCHAR(255)	NOT NULL
venue_id	INT(11)	NULL
venue_name	VARCHAR(255)	NULL
venue_street	VARCHAR(255)	NULL
venue_postcode	VARCHAR(16)	NULL
venue_city	VARCHAR(128)	NULL
venue_country	VARCHAR(100)	NULL
venue_longitude	FLOAT	NULL
venue_latitude	FLOAT	NULL
venue_url	VARCHAR(255)	NOT NULL
lastfm_url	VARCHAR(255)	NOT NULL
startdate	INT(11)	NULL
attendance	INT(11)	NOT NULL
reviews	INT(11)	NOT NULL

Figure 4.3.: Database schema for the ReSEED dataset

key *flickr_picture_id*. The columns contain all textual and numerical data as described in detail in Section 4.3 but with the exception of the tags. Table `ReSEED_PICTURES_TAGS` provides an association between the pictures and all assigned tags. All machine tags have been removed from the set of tags.

The associations between the picture documents and the corresponding events are stored in the table `ReSEED_PICTURES_EVENTS`. In addition to the aggregated event ID as described in Section 4.2, the individual event IDs from the original online event calendar providers Upcoming and last.fm are also stored. With the help of these references the original data from the calendar providers can be retrieved.

An explanation of the data included in the event calendar data tables `ReSEED_EVENT_UPCOMING` as well as `ReSEED_EVENT_LASTFM` is given in Table 4.1 together with more details in Section 4.2.2.

4.4 Applications of the Dataset

The ReSEED dataset has not only been created for the testing and evaluation of our event-based clustering framework presented later in this work but also to support other researchers in the field of social event detection with a freely available dataset allowing for a comparison of results from different approaches. The dataset has already been used in the Social Event Detection task, and due to its nature it is also useful for other research questions in the area of event detection and searching.

4.4.1 MediaEval 2013

As already mentioned in Section 3.7, the dataset was already successfully used in the 2013 edition of the Social Event Detection (SED) task at MediaEval. There were eleven teams with about 65 people participating in the task. The task description for the challenge involving the dataset as proposed by us was to "produce a complete clustering of the image dataset according to events".

The task thus consisted in the automatic induction of event-related clusters for all pictures in the dataset, determining the number of events in the dataset automatically. This is comparable with the task explored in this thesis with the difference that the stream-based scenario was optional.

The challenge had one required run which involved using only the metadata. We have forbidden the use of additional data for this run (e. g. visual information from the images) making all approaches comparable with each other. For the other runs it was allowed to use generic external resources like Wikipedia, WordNet, or visual concept detectors trained on other data. However, it was not allowed to use external data that was directly related to the individual images included in the dataset, such as machine tags.

4.4.2 Further Applications

A different challenge of the MediaEval 2013 task was a combined event identification and event category detection task. It consisted of the classification of pictures into one set of pictures depicting an event and another set not depicting any event (non-event). The pictures in the first set then should be classified again into a fixed set of eight event types: concert, conference, exhibition, fashion, protest, sports, theater & dance, and others. This task thus represents a standard classification task that can be addressed using standard supervised classification approaches and exploiting both visual features and metadata making it interesting for the use with ReSEED.

Another task consists of the enrichment of the dataset with other types of social media information items coming from other sources than Flickr; this might be Twitter or other social network sites like Facebook. The advantage here is that the search for additional material for the labeled classes is less expensive than the creation of new labels of yet unknown classes.

Without any modification, ReSEED can further be used in all kind of tasks related to the semantic search and information retrieval of pictures and/or social media events.

4.4.3 Evaluation Proposal for Comparability

To make the approaches evaluated with the help of this dataset comparable, we have split the dataset into a training and an evaluation part. We also propose certain evaluation measures to ensure good comparability.

Dataset Splits for Training and Test

ReSEED has been split into two parts. We declared 70 % of the dataset to constitute the training set; the rest is supposed to be used for evaluation purposes. The splits were made without the overlap of any events. To ensure that both parts include similar data, i. e. in regard to the number of events, the size of events, and the upload timestamps, the splits have been created with the following steps:

1. All events of the dataset are ordered by their number of pictures included in the event.
2. Starting with the largest event, the events are assigned to split 1 through 10 in descending order of their event size. E. g. the largest event is assigned to split 1, the second largest to split 2, the tenth largest to split 10, the eleventh largest to split 1, and so on.
3. If a split is full, i. e. it contains exactly one tenth of the overall amount of pictures (which is in our case 43,737), we skip that split. As the number of events including only two or one picture is large, there are no problems with events not fitting into a split (see also 4.2).
4. The final 70/30 parts are then created by merging splits 4–10 as well as splits 1–3 to the training and test set, respectively.

Evaluation Measures

For evaluation we propose to use different evaluation measures in order to compare the results of the produced clustering to the ground truth information that has been compiled from the event calendars. All evaluation measures return values in the range of [0..1] where higher values indicate a better agreement with the gold standard. The three evaluation measures are the following:

- *F_1-Score*: We propose F_1-Score to be used as the main evaluation measure. In the evaluation script provided together with the dataset, the micro-averaged version was used to calculate the harmonic mean of precision and recall. This measures the appropriateness of the event clusters.
- *NMI*: Normalized Mutual Information (NMI) is used to compute the overlap between the clusters.

- *Divergence from a Random Baseline*: This method indicates how much the results diverge from a random baseline as described in Vries, Geva, and Trotman [VGT12]. This is proposed to be used as a sanity check.

All the measurements are implemented in an evaluation script delivered together with the dataset[8] so that algorithms can be compared easily to other results. An exact definition of each of these measures is given in Section 6.1.

[8]http://doi.org/10.4119/unibi/citec.2014.10

PART II

Supervised Single-Pass Clustering with the Event-based Stream Classification Framework

CHAPTER 5

System Description of the Stream Classification Framework for a Single-Pass Setting

In the previous chapter, we presented a real-world and non-toy social media dataset with images. We made sure that the documents included in the dataset belong to exactly one event. This dataset is prototypical for a real-world application where the aim is to identify and detect distinct events within the set. We conduct this process in order to prepare and organize the social media documents—in our case pictures—in a way that enables people to better browse and search the data more naturally.

Even though the dataset can be used as a fixed set where all items within the dataset are accessible at every moment, our intent is to use it in a way where the data is arriving continuously. This seems to be natural as in real-world applications the documents also arrive first after they have been posted or uploaded to the social media site. This is in line with the perspective of the organizations hosting a social media application from where the uploaded postings, messages, pictures, etc. can be seen as an exponentially growing and never-ending stream of data. We denote such a scenario as *stream setting*. In such a setting, it is not possible during the process to know which document arrives next before it actually appears in the data stream. For that reason, the documents in the dataset are ordered by the timestamp denoting the time when they have been uploaded to the Flickr community site.

Given the above explanations, our aim is to develop a system allowing for an automated clustering of such a data stream of social media documents. Such clustering could serve as preparation for a better organization of the data. It seems clear that there is an impending need for such an automatic technique that performs the assignment of a social media document which has been newly uploaded to some social media site like Flickr to its corresponding event or, if it does not exist yet, creates a new event to which future data points can be assigned. In line with previous work we refer to this problem as the *social event detection problem*. The problem statement of this task is as follows:

Problem 5.1 Given a data stream of social media documents where each document is associated to an unknown event, our goal is to discover these events and identify which documents are associated to the same event. ■

When considering the task of *social event detection*, we are thus faced with the challenge of classifying a massive and never-ending stream of data into their corresponding events. We therefore cast the problem as a classification problem. There are at least three challenges involved in developing a system that can cluster data streams as originating in social media applications into event categories. The first challenge is to deal with the massive amount of data arriving per minute. The second challenge is to classify documents into potentially millions of events. And the third challenge is to deal with the fact that the set of events to which documents are assigned is constantly growing.

In an ideal case, each created event cluster corresponds to exactly one real-world social event. In addition, such a cluster consists of all the social media documents which are related to that event. The algorithm of choice should be scalable to deal with the vastness of social media data and should be able to handle the massive amounts of data arriving in short time. Due to the growing nature of the target categories—social media sites are constantly growing and evolving—standard supervised learning techniques assuming the target categories are fixed are less suited for the task. For example, more traditional clustering algorithms like K-Means or Expectation–Maximization require knowledge about the number of categories beforehand. This leads us to develop a framework overcoming these limitations and making a clustering of social media documents possible.

Therefore, in this chapter, we introduce our *Event-based Stream Classification Framework* which we developed for event clustering using a single-pass strategy. We will show that the system presented in this dissertation is indeed capable of clustering a social media documents stream into the corresponding events and is able to process the social media dataset presented before addressing the above mentioned three challenges. We frame the problem as a stream data classification problem leading to a supervised classification framework. Each new data point uploaded to the system is either classified into one of the existing events or a new event is created in the system.

5.1 Problem Statement

The problem of social event detection in social media can be formulated as a supervised classification problem. The underlying similarity function and other parameters of the clustering algorithm are optimized using labeled data.

Let E be the set of all event clusters. In principle we need a function a_t defined as follows:

$$a_t : D \to E_t \tag{5.1}$$

This function a_t assigns a document $d \in D$ to some of the events E available at time t. In our case the events E_t are the clusters. As the number of documents and the number of event clusters is unknown, we aim to depend on a similarity function sim for our decision. If a new event detection function using a feature vector $\vec{v}_{new}(d)$ does not decide on the creation of a new event, this function sim assigns a document to the event cluster maximizing the similarity:

Definition 5.1.1 — Assignment function

$$a_t(d) = \arg\max_{e \in E_t} sim(d, e) \tag{5.2}$$

The desired similarity function to be learned thus has the following form:

$$sim : D \times E_t \to [0..1] \tag{5.3}$$

A central question is which model to assume for such a function. In our approach we assume a linear model. Therefore, we use features such as time, location, tags, titles, etc. The features correspond essentially to simple similarity measures for different dimensions. The actual similarity between a document and an event is calculated using the following linear model:

Definition 5.1.2 — Similarity function

$$sim(d, e) = \vec{w}^T \vec{v}_{sim}(d, e) \tag{5.4}$$

In Equation (5.4), $\vec{w}$ denotes a weight vector and $\vec{v}_{sim}$ is the feature vector of a document-event pair. Therefore, the problem is to optimize the weight vector $\vec{w}$—and thus the function sim—for the task of assigning documents to the matching event.

In order to formalize the idea, let us introduce some terminology. Let d_i be a data point to be assigned to some cluster. Let $event(d_i)$ be the correct cluster for d_i according to our labeled training data. We define the following functions:

- A function *centroid* assigning each event a centroid vector averaging over all data items belonging to this event: $centroid : E \to \mathbb{R}^n$.
- A function *event* assigning each document d_i to its corresponding event: $event : d_i \to E$. This function is only used for evaluation purposes.
- A function defining the extension *ext* of a certain event $e \in E$ consisting of all data items belonging to this event: $ext : E \to \mathbb{P}(D)$.

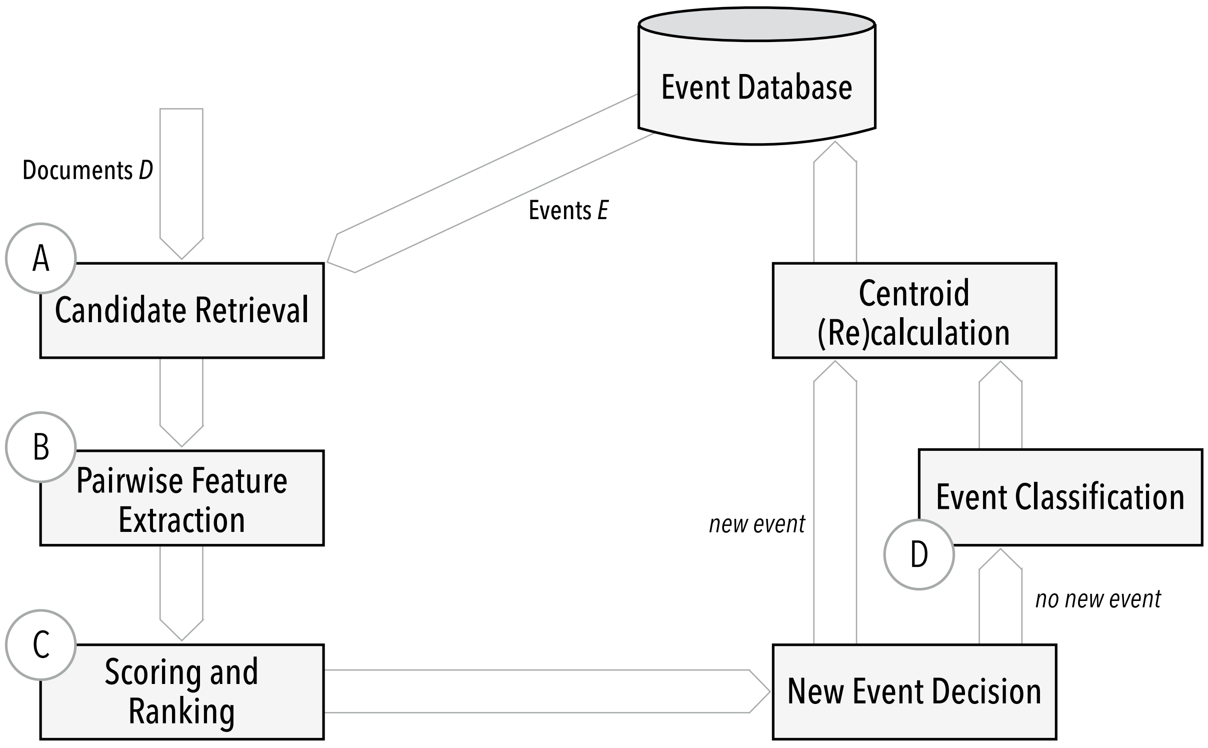


Figure 5.1.: Overview of the event clustering framework system

5.2 Overview of the Clustering Framework

In the following we demonstrate how our framework can handle an incoming data stream of documents D in order to cluster these documents into their corresponding events E. Our system processes such an incoming stream of documents D in several steps. As also depicted graphically in Figure 5.1, for each incoming document $d \in D$ the following steps are conducted:

1. *Candidate Retrieval*: A set E of k events that d is likely to belong to are retrieved from the event database (see also Section 3.5.3).
2. *Scoring and Ranking*: For each of these candidates $e \in E$ retrieved in the previous step, we compute the probability $P(e|d)$ of d belonging to E. All candidates are then ranked according to that probability. Let e_{max} or e_1 denote the top scoring candidate.
3. *New Event Decision Probability*: Given the ranked list of candidates, the probability, that d belongs to a new event, $P_{new}(d)$, or that it belongs to the first event in the list, $P_{belongs\_to\_top\_cand}(d)$, is computed. We assume that these are the only two options, i. e. $P_{new}(d) + P_{belongs\_to\_top\_cand}(d) = 1$.
4. *New Event Detection*: If $P_{new}(e) > \theta_n$, for a given threshold θ_n, a new event e' is created and d is assigned to this newly created event: $\vec{e'} := \vec{d}$.

Algorithm 5.1: Stream-based clustering into events

```
Input   : A stream of social media documents $D$
Output  : Documents clustered by their event
foreach $d \in D$ do
    $Top_k(d) \leftarrow$ retrieve a ranked list of promising event candidates to which $d$ could belong;
    foreach $e \in Top_k(d)$ do
        compute $P(e|d)$;                        // the probability that $d$ belongs to $e$
    end
    $e_{max} \leftarrow \arg\max_{e' \in Top_k(d)} P(e'|d)$;
    compute $P_{new}(d)$;                 // the probability that $d$ belongs to a new event
    if $P_{new}(d) > \theta_n$ then
        create new event $e'$;
        $e' = \{d\}$;
        $\vec{e'} = \vec{d}$;
    else
        $e_{max} = e_{max} \cup \{d\}$;
        recompute $\vec{e}_{max}$;
    end
end
```

5. *Event Classification*: Otherwise, d is assigned to e_{max}; the centroid $\vec{e}_{max}$ is recomputed.

The above procedure is illustrated more formally by the pseudocode in Algorithm 5.1. There are some crucial aspects of our approach which we describe in the following. We are pointing to the relevant sections where the corresponding components are described in more detail.

- **(A)** The candidate retrieval step relies on a set of queries that build on appropriate inverted indices (for text data like tags) as well as on timestamps to retrieve a set of promising candidates. This step is crucial to keep the approach scalable and to avoid having to process all events stored in the database. This step is described in more detail in Section 5.3.
- **(B)** As feature representation, we compute a number of similarities between a document d and each event e from the list of top-k retrieved events. This is described in more detail in Section 5.4.
- **(C)** The above mentioned probabilities $P(e|d)$ and $P_{new}(d)$ are computed using different machine learning approaches like Support Vector Machines or Decision Trees trained on an appropriate training dataset. Below, we describe in more detail how these approaches are trained and how the probabilities are computed. Each model is trained on a separate training data split. This is described in more detail in Section 5.5.
- **(D)** An important parameter is the hyperparameter θ_n, which essentially determines whether a new event is created or the new data item is assigned to an existing event in the

database. This hyperparameter is also tuned on the training data split. This important step for the new event decision is described further in Section 5.6

The different steps mentioned above are all described in detail in the following sections.

5.3 Candidate Retrieval Strategies

In stream-based and large-scale clustering the most crucial point is scalability. However, it is challenging to develop techniques which scale to the amounts of data available in social media applications. Our goal is to support the classification of incoming documents into events in real time.

A very effective technique to scale up and achieve real-time behavior is to use a suitable *candidate retrieval strategy*. The idea is to reduce the set of events that are considered for every incoming document. As there might be billions of events in the database, it is not feasible to scan all these events to compute the likelihood that the incoming document belongs to it. For this purpose, candidate retrieval or so-called *blocking* strategies are typically considered [BCC03], which massively reduce the amount of pairs (in our case these are pairs of document and event) to be considered. The reason for us to reduce the number of candidates is that the predictions of our classifier need to be calculated for each candidate event, i. e. the probability $P(d|e)$ that the documents belongs to this event. This computation is linear in the number of events, which can be prohibitive if the number of events is large, for example in the region of millions or even billions.

When using a candidate retrieval technique, a crucial issue that arises is to ensure that the blocker is neither too strict, thus filtering too much, nor too lenient, possibly passing on too much noise to the classification algorithm. A certain candidate retrieval strategy might thus be efficient from a computational point of view but too strict, thus having an overall detrimental effect on the overall performance of the main algorithm.

Therefore, we require an event candidate retrieval strategy c_i that, ideally, i) retrieves the correct event and minimizes the overall number of retrieved events and thus ii) filters out as many of the irrelevant events as possible in order to reduce the computational cost of post-processing the retrieved events.

5.3.1 Measurements for Performance and Effectiveness

In this section we define criteria on how we can assess different candidate retrieval strategies. We are interested in especially two factors: i) effectiveness and ii) performance.

The effectiveness is used to analyze how many data points have to be retrieved in order to include the required data points. We measure this effectiveness of a candidate retrieval strategy c_i at a number k of retrieved events as in Equation (5.5):

Definition 5.3.1 — Effectiveness of a candidate retrieval strategy

$$\text{Effectiveness}(c_i, k) = \frac{|\{d \mid \text{event}(d) \in \text{top}_k(c_i)\}|}{|D|} \tag{5.5}$$

Where $\text{event}(d)$ is the correct event for a document d according to the gold standard, and $\text{top}_k(c_i)$ is the set of top k events retrieved by the candidate retrieval strategy c_i.

The performance measurement is used to get information about the retrieval time and thus the computational cost. To make the values comparable we measure the time needed to retrieve all events and set this to be the reference.

5.3.2 Candidate Retrieval Strategies

For our experiments, we take a look at several candidate retrieval strategies which we compare to each other using the measurements defined above. In particular we compare the following strategies.

1. **k-nearest by capture time**: By using this strategy, we retrieve those k events with the lowest temporal distance to the document. For this, we order all events e_i in the event database by $\Delta(\text{time}(d), \text{time}(e_i))$ and then return the top k events in this ordered list.

2. **k-nearest by upload time**: We follow the same strategy as in *k-nearest by capture time* but use the upload timestamps.

3. **Geo-Blocker**: We define a fixed window of $1.0° \times 1.0°$ and select the top-k events out of this window, i. e. we retrieve all events such that $\text{latitude}(d) - 1° < \text{latitude}(e) < \text{latitude}(d) + 1°$ and $\text{longitude}(d) - 1° < \text{longitude}(e) < \text{longitude}(d) + 1°$. This window roughly corresponds to an area of 50-60 square kilometers in Europe and the U. S. and to an area of 110 square kilometers in the vicinity of the equator. The candidates are ordered by simple Euclidean distance using the longitude and latitude values.

4. **Tag-TFIDF**: We score each event by summing up the TF-IDF values of every tag which the document and the event share and return k events having the highest scores.

5. **Title-TFIDF**: This is the same as *Tag-TFIDF* but using the tokens in the title instead of tags.

6. **Description-TFIDF**: This is the same as *Tag-TFIDF* but using the tokens in the description.

7. **Uniform Combination**: We retrieve the same number of $\frac{k}{6}$ (with $k \bmod 6 = 0$) events for each of the candidate retrieval strategies $c_1, \cdots, c_6$ described above.

8. **Optimal Combination**: We determine the optimal combination of the number of candidates to be retrieved by each candidate retrieval strategy empirically. We thus retrieve a number $k(c_i)$ of events that is specific for each candidate retrieval strategy. The optimal parameters are computed by exhaustive search on the training set with the goal of maximizing the effectiveness (see above) of the candidate retrieval strategy.

5.4 Pairwise Feature Extraction

In this section we describe the specific features that are computed on the basis of the metadata described above to represent an image. A pair of a document and a candidate event is described in terms of a vector of several features that describe the match between the document and the event. To create this vector we define similarity measures that are used to compare two pictures along each of these feature dimensions.

In particular, we define nine similarity measures exploiting the information sources which we present in the following. We also discuss alternatives and the advantages of the chosen measurement.

5.4.1 Temporal Features

Temporal information refers to the time point the document was created (e. g. by taking a picture with a camera), which is typically when the event took place or the time when the user uploaded the document to a social media site. Each document as well as each event have a timestamp assigned. Let $time(d)$ denote a timestamp assigned to a document and $time(e)$ denote a timestamp assigned to an event.

To define a similarity measurement for temporal information, we start with some basic assumptions.

- It is possible to measure the distance between the two timestamps by calculating the delta: $\Delta time = |time(d) - time(e)|$. This distance can then be interpreted in terms of similarity.
- The similarity of two data points is maximal if both timestamps are exactly similar (therefore, $\Delta time = 0$).
- The more apart the two timestamps are (the greater $\Delta time$), the less similar are the two data points.
- Without the definition of any boundaries, a total dissimilarity would be only possible if the distance $\Delta time$ is infinite.

These assumptions in mind leads us to a similarity measure $sim_{\text{time}}(d, e)$ defined as what follows:

$$sim_{\text{time}}(d, e) = \begin{cases} 1.0 - \dfrac{\Delta time}{\Delta year} & \text{if } \Delta time \leq \Delta year \\ 0.0 & \text{else} \end{cases} \tag{5.6}$$

In Equation (5.6), $\Delta year$ is defined as the distance between two timestamps which are exactly one year apart. The similarity using this measurement is in the range of [0..1]. A similarity of 1.0 means that the timestamp of the two data points are the same. A value of 0.0 is interpreted as totally dissimilar; this is the case if two timestamps are more apart than one year. The maximal distance of one year has been chosen to ensure two things: a) we want to include events of all kinds of length, and b) we do not want to include recurring events such as Christmas as they are seen as distinct events by our definition of an event.

A disadvantage of the measure given in Equation (5.6) is its strict linearity, which does not reflect that there are differences in the importance of relative distances depending on the absolute distance. For example, it does not make a large difference if two data points have a distance of 300 or 301 days, but we care if two data points have a distance of 1 hour instead of 25 hours. There is a difference of 24 hours in both cases but the latter case is of more interest than the first one. Therefore the measure is changed to include a logarithmic operation. Our final similarity measure thus looks as follows:

Definition 5.4.1 — Similarity of two timestamps

$$sim_{\text{time}}(d, e) = \begin{cases} 1.0 - \dfrac{log(\Delta time)}{log(\Delta year)} = 1.0 - \dfrac{log(|time(d) - time(e)|)}{log(y)} & \text{if } \Delta time \leq \Delta year \\ 0.0 & \text{else} \end{cases} \tag{5.7}$$

Using the ReSEED dataset, $time(d)$ and $time(e)$ are timestamps represented as integer values denoting the number of minutes elapsed since the Unix epoch, and y is the number of minutes of one year.

This yields two similarity measures:

- $sim_{\text{capture}}(d, e)$, calculated on the basis of capture time.
- $sim_{\text{upload}}(d, e)$, calculated on the basis of upload time.

5.4.2 Geographical Features

The geographical information locates the place where the document was created; this is the location of an event in terms of latitude and longitude. This information is typically added to the image with the aid of a GPS device connected to the camera.

As with the time information, we have to determine the distance between two geographic location points. Therefore, we incorporate the latitude and longitude of both points. Initially, as we did not need to measure an exact distance between two points but only need a coarse estimate to judge their similarity, we aimed to calculate the Euclidean distance between two latitude-longitude pairs in order to get a computationally cheap distance metric. Unfortunately, the distance between two points varies a lot depending on the latitude; the real distance of two points at the equator and two points in the polar region, each with 1° difference, are not comparable. As a consequence, we use a distance function taking this characteristic of latitude/longitude values into account. It is possible to use two different measures with no noteworthy difference in computational complexity; the first uses the direct distance through Earth, the second uses the correct distance along the surface.

In our approach we incorporate the haversine formula to determine the great-circle distance $H(L_1, L_2)$ between two locations L_1 and L_2 on Earth. Each location L_n is defined by its coordinates denoted by latitude lat_n and longitude lon_n. The definition of the haversine formula is given as follows:

Definition 5.4.2 — Haversine formula

$$H(L_1, L_2) = 2 \cdot \arctan^2(\sqrt{\phi}, \sqrt{1-\phi}) \tag{5.8}$$

$$\phi = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2\left(\frac{\Delta lon}{2}\right) \tag{5.9}$$

$$\Delta lat = |lat_2 - lat_1|, \quad \Delta lon = |lon_2 - lon_1| \tag{5.10}$$

The distances returned by the above formula are in the range [0..1]. The multiplication with the equatorial distance as in the original definition of the haversine formula is omitted, as a distance factor between 0 and 1 is sufficient for our task.

Finally, the geographic similarity is defined as follows:

Definition 5.4.3 — Similarity of two geographic locations

$$sim_{\text{geo}}(d, e) = 1.0 - H(L_1, L_2) \tag{5.11}$$

5.4.3 Textual Features

Besides the time and geographic features, there is a lot of information manually added by users in textual form. These are, for example, tags, a title, a description, etc.

To compare textual information, there are two kinds of distance metrics which can be used on strings: term-based metrics and edit-distance metrics. Using the first, the distance depends on a set of words which are contained in both documents to be compared. An example here is the Jaccard distance or the cosine similarity of a TF-IDF vector. The latter method denotes the distance by the shortest sequence of edit commands that is able to transform the text string from document d into the one from document e. An example here is the Levenshtein distance.

In our scenario it seems to be natural to use a term-based metric. The idea behind this is the assumption that two documents which do share the same terms (think about the assigned keywords or tags) are more similar to each other than those documents which do not. However, we do not only want to differentiate between the two cases *has same terms* and *does not have same terms*, but we aim to use a linear metric.

Before we decide on a distance metric, we have to introduce an algebraic model capable of representing textual information as a vector, the so-called *Vector Space Model.* Such a vector could represent either the absence or presence of terms in a document, or the importance of terms. For example because of stop words (words present in almost all documents), the importance of the terms should be taken into account. Therefore, we use a weight function $D \times T \rightarrow \mathcal{R}$ for each term t in document d, namely *TF-IDF.*

The TF-IDF of each term in the document is defined as follows:

Definition 5.4.4 — TF-IDF

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \text{IDF}(t) \tag{5.12}$$

The term frequency $\text{TF}(t,d)$ is defined as the absolute number of appearances of term t in document d. The inverse document frequency $\text{IDF}(t)$ is defined as follows:

Definition 5.4.5 — Inverse Document Frequency (IDF)

$$\text{IDF}(t) = log\frac{n}{\text{DF}(t)} \tag{5.13}$$

Here, n denotes the overall number of documents in the corpus and $\text{DF}(t)$ is the document frequency, which is the absolute number of documents where the term t appears.

Having the weights for each term, a document can then be represented as a vector of term weights. Now, it is possible to calculate the similarity between the two documents in the vector space using cosine similarity. The cosine $\cos(\theta)$ of two vectors $\vec{v_d}$ and $\vec{v_e}$ is the similarity between the two documents and can be derived using the scalar product between the vectors:

$$\vec{v_d} \cdot \vec{v_e} = ||\vec{v_d}|| \cdot ||\vec{v_e}|| \cdot \cos(\theta) \tag{5.14}$$

From this it follows that $\cos(\theta)$ is defined as follows:

$$\cos(\theta) = \frac{\vec{v_d} \cdot \vec{v_e}}{||\vec{v_d}|| \cdot ||\vec{v_e}||} = \frac{\sum_{i=1}^{n} \vec{v_d}^{(i)} \times \vec{v_e}^{(i)}}{\sqrt{\sum_{i=1}^{n} \left(\vec{v_d}^{(i)}\right)^2} \times \sqrt{\sum_{i=1}^{n} \left(\vec{v_e}^{(i)}\right)^2}} \tag{5.15}$$

The final similarity measure for textual data sim_{tags}, sim_{title}, and $sim_{\text{description}}$ with an output between 0.0 and 1.0 is thus defined as follows:

Definition 5.4.6 — Similarity of two documents regarding their text tokens

$$sim_{\text{text}}(d,e) = \frac{\sum_{t} \text{TF-IDF}(t,d) \times \text{TF-IDF}(t,e)}{\sqrt{\sum_{t} \text{TF-IDF}(t,d)^2} \times \sqrt{\sum_{t} \text{TF-IDF}(t,e)^2}} \tag{5.16}$$

5.4.4 Document-Event Similarity Vector

Overall, using the features described above, a feature vector for a pair of document d and event e looks as follows:

$$\vec{v}_{sim}(d,e) = \begin{pmatrix} sim_{\text{upload}}(d,e) \\ sim_{\text{capture}}(d,e) \\ sim_{\text{geo}}(d,e) \\ sim_{\text{tags}}(d,e) \\ sim_{\text{title}}(d,e) \\ sim_{\text{description}}(d,e) \end{pmatrix} \tag{5.17}$$

In this vector, $sim_{\text{upload}}(d,e)$ and $sim_{\text{capture}}(d,e)$ denote the similarities between the upload time and capture time of a document-event pair, $sim_{\text{geo}}(d,e)$ is the geographic similarity, and $sim_{\text{tags}}(d,e)$, $sim_{\text{title}}(d,e)$, and $sim_{\text{description}}(d,e)$ denote the similarities for the tokenized text features tags, title, and description, respectively.

5.5 Scoring and Ranking – Learning Similarity Functions

For each incoming document d, we compute the likelihood $P(e|d)$ that it belongs to a given event e. We then order the events retrieved by the candidate retrieval step by decreasing probability. The likelihood that document d belongs to event e is calculated using a classification algorithm. In this thesis we use two different classifiers to learn the similarity function: *Support Vector Machines (SVM)* and *Decision Trees.*

To this end we rely on an appropriate training dataset to discriminate between pairs of document and corresponding event and pairs of documents that do not belong to the given event. The first are our positive examples while the latter are the negative ones. We use different numbers of examples for each of these classes to train the classifiers and compute the probability $P(e|d)$ as the probability that the pair (d, e)—as described by the above mentioned vector of similarities—belongs to the positive class, i. e. $P(e|d) = P(\text{positive}|\vec{v}_{sim}(e, d))$.

In the following we formulate the problem using a Support Vector Machine and a Decision Tree. We take a look at "standard" SVMs as well as Ranking SVMs.

5.5.1 Problem Formulation using a Support Vector Machine

Using a SVM as a classifier, we aim to determine a hyperplane $\langle \vec{w}, sim(\vec{d}, e)\rangle + b = 0$ which separates two classes. This is done in a way so that the margin to this hyperplane is maximized. The margin is the region enclosed by the two additional hyperplanes described by the following equations:

$$\langle \vec{w}, \vec{v}_{sim}(d, e)\rangle + b = 1 \tag{5.18}$$

$$\langle \vec{w}, \vec{v}_{sim}(d, e)\rangle + b = -1 \tag{5.19}$$

It is rarely possible to achieve a perfect separation between the positive and negative examples. As a consequence, a classification error must be allowed. Thus, the goal is to allow a soft margin determining a hyperplane which minimizes errors and maximizes the margin that separates the data. According to Cortes and Vapnik [CV95], this problem can be specified as in the following equations:

$$min_{w,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{\ell} \xi_i \tag{5.20}$$

$$y_i(\langle w, \Phi(x_i)\rangle + b) \geq 1 - \xi_i, \quad \xi_x \geq 0 \tag{5.21}$$

As usual in classification, $\Phi(x_i)$ denotes some kernel, helping to avoid the explicit mapping needed to learn a nonlinear decision boundary; this can be the dot product in the simplest case. The variable C is a constant defined by the user which allows to trade off the margin size against the training error; ξ_i are slack variables and $y_i \in \{-1, 1\}$ is the class label.

In our scenario, the training examples x_i are defined as follows:

$$x_i = \begin{cases} ((d_i, e), -1) & \text{if } e' \notin event(d_i) \\ ((d_i, e), +1) & \text{else} \end{cases} \tag{5.22}$$

We assume a functional margin of 1 as typically used in Support Vector Machines. Therefore, the weight vector $\vec{w}$ needs to fulfill the following conditions:

$$\begin{aligned} \forall d_i \quad & \langle \vec{w}, \Phi(d_i, event(d_i)) \rangle > 1(-\xi_i) \\ \forall d_i \forall e' \neq event(d_i) \quad & \langle \vec{w}, \Phi(d_i, e') \rangle < -1(+\xi_i) \end{aligned} \tag{5.23}$$

Such a function is applied to compare a new document to the centroid of every cluster constructed so far and retrieved by the candidate retrieval step. The new data point is assigned to the cluster maximizing this similarity, provided it fulfills criteria being classified as belonging to an existing cluster by the new event detection step (see next section). If a new cluster e' is created at time t, then $E(t+1) = E(t) \cup e'$. Learning the similarity function is thus formulated as a standard classification problem, separating those pairs of documents that belong to the same event from those that do not.

Problem Formulation as a Ranking SVM Problem

According to the formulation of the problem to use the similarity function to determine the cluster having the highest similarity to the given data point, we can see the problem in fact as a ranking problem. In this case, the task is to rank all clusters according to their similarity to the data point. This would provide us with an alternative formulation of the problem, i. e. the one of learning a similarity function that always assigns a higher value to the correct cluster compared to all other clusters.

Therefore, the task is to learn a similarity function that satisfies the condition in (5.24) or the equivalent condition in (5.25).

$$\forall d_i \, \forall e' \notin event(d_i) \quad sim(d_i, event(d_i)) > sim(d_i, e') \tag{5.24}$$

$$\forall d_i \, \forall e' \notin event(d_i) \quad sim(d_i, event(d_i)) - sim(d_i, e') > 0 \tag{5.25}$$

Assuming that we have a linear model for sim and the weight vector is denoted by $\vec{w}$, we get the following condition:

$$\forall d_i \, \forall e' \notin e(d_i) \quad \vec{w} \left(\vec{v}_{sim}(d_i, event(d_i)) - \vec{v}_{sim}(d_1, e') \right) > 0 \tag{5.26}$$

Note that this problem is essentially equivalent to the optimization problem solved by the ranking SVMs of Joachims [Joa02]. Assuming a margin of 1 we would get the following equivalent formulation, which is the one used by Joachims:

$$\forall d_i \, \forall e' \notin event(d_i) \quad \vec{w} \left(\vec{v}_{sim}(d_i, event(d_i)) - \vec{v}_{sim}(d_1, e') \right) > 1 - \xi_i \tag{5.27}$$

Given this formulation of the problem of learning an adequate similarity as a learning to rank problem, we can apply the Ranking SVM of Joachims in an off-the-shelf manner directly to the problem of learning a suitable similarity function for the event detection problem.

Clearly, learning a $\vec{w}$ that fulfills the condition in (5.27) also implies finding a $\vec{w}$ that fulfills the conditions (5.20) and (5.21). In some sense, the conditions in (5.21) might even be too restrictive for the problem at hand. In this sense, we might be solving a more difficult problem than the one we indeed have to solve when using the standard SVM criteria.

Calculation of Probabilities for Support Vector Machines

Usually, a Support Vector Machine calculates a decisions function $f(x)$ in a way that the label y_i of any test example x can be predicted using the function $\operatorname{sgn}(f(x))$. For our classification, we aim to use a posterior class probability $P(y=1|x)$ instead of the direct prediction of the label. A solution to achieve such an output has been proposed by Platt [Pla99] by approximating the probability by a sigmoid function:

$$P(y=1|x) \approx P_{A,B}(f) \equiv \frac{1}{1+e^{Af+B}} \tag{5.28}$$

The maximum likelihood estimation from a training set (f_i, y_i) is used to fit the parameters A, B. The best parameter setting can be determined solving the regularized maximum likelihood problem:

$$\min -\sum_i \left(t_i \log(p_i) + (1-t_i)\log(1-p_i)\right) \tag{5.29}$$

where $p_i = P_{A,B}(f_i)$ and t_i depend on whether y_i is positive or not:

$$t_{i+} = \frac{N_+ + 1}{N_+ + 2}, t_i- = \frac{1}{N_- + 2} \tag{5.30}$$

Using this in conjunction with a SVM therefore enables us to calculate the likelihood of whether the document belongs to a certain class.

5.5.2 Problem Formulation as a Decision Tree Classification Problem

Using a Decision Tree, we aim to learn simple decision rules based on the features of each document-event pair cascaded in a binary tree, deciding whether they belong to the same class or not. The space is partitioned recursively in each node by the Decision Tree.

Let us represent the data at a Decision Tree node m by S. This data S is partitioned into S_{left} and S_{right} subsets for each candidate split $c = (sim_j(d,e), \Theta_m)$ where $sim_j(d,e)$ is one feature of the similarity vector $\vec{v}_{sim}(d,e)$ and Θ_m is a threshold:

$$S_{\text{left}}(c) = (x,y)|(x_{sim_j(d,e)} \leq \Theta_m) \tag{5.31}$$

$$S_{\text{right}}(c) = S \setminus S_{\text{left}}(c) = (x,y)|(x_{sim_j(d,e)} > \Theta_m) \tag{5.32}$$

We calculate the impurity of the subsets using the Gini index $I_G(f_m)$. The probabilities or relative frequencies (f_{m_i}) of items in each class i at a node m are squared and summed up resulting in the Gini index defined as follows:

$$I_G(f_m) = 1 - \left(\sum_{i \in (-1,1)} (f_{m_i})^2 \right) \tag{5.33}$$

Given this impurity function, the information gain of a partitioning $G(S,c)$ can be computed as follows:

$$G(S,c) = I_G(S) - \left(\frac{n_{\text{left}}}{N_m} \cdot I_G\left(S_{\text{left}}(c)\right) + \frac{n_{\text{right}}}{N_m} \cdot I_G\left(S_{\text{right}}(c)\right) \right) \tag{5.34}$$

In the above equation N_m denotes the number of data point observations at node m. The variables n_{left} and n_{right} are the number of observations in the left or right subset.

Finally, the goal of the Decision Tree algorithm is to find the partitioning $\dot{c}$ maximizing the information gain within all created partitions:

$$\dot{c} = \arg\max_{c} G(S,c) \tag{5.35}$$

This is equal to minimize the impurity of the subsets:

$$\dot{c}^* = \arg\min_{c} \left(\frac{n_{\text{left}}}{N_m} \cdot I_G\left(S_{\text{left}}(c)\right) + \frac{n_{\text{right}}}{N_m} \cdot I_G\left(S_{\text{right}}(c)\right) \right) \tag{5.36}$$

Using recursion the steps (5.31) through (5.36) have to be repeated for all subsets $S_{\text{left}}(\dot{c})$ and $S_{\text{right}}(\dot{c})$ until we have reached the maximal allowed tree depth, $N_m = 1$ or $N_m < \text{min}_{\text{datapoints}}$.

5.6 New Event Detection

Once the candidates are ranked by the likelihood of whether d belongs to these candidates, we are faced with the important question whether the document d belongs to the top-ranked event or rather to a new event which has to be created. This is what we refer to as the *new event detection* problem.

It is not clear if the top-scoring candidate belongs to a new event or not. It might actually represent an event that d is not related to. Therefore, we need another decision function that decides on the question whether the document d shall be assigned to the top-scoring candidate or to a newly created event.

For this purpose we employ a different classifier trained on an appropriate training dataset. As in the scoring and ranking problem we employ a standard Support Vector Machine as well as a Decision Tree which we compare with each other. Both are trained with examples in which the document belongs to the top-scoring event and examples in which the document belongs to a new event; the first are our positive examples, the latter the negative ones.

We have taken the following features into account:

- **Best score** (max): This is $P(e_1|d)$, i. e. the probability that d belongs to the top-scoring event.
- **Worst score** (min): The probability $P(e_k|d)$—this is the probability that d belongs to the k-th ranked event, where k is the number of candidates retrieved by the candidate retrieval step.
- **Average score** (avg): This feature is the averaged probability of the top-k most likely events: $\frac{1}{k}\sum_{i=1}^{k} P(e_i|d)$.
- **Standard deviation of score** (stddev): Here, we use the standard deviation of the top-k most likely events.
- **Maximum capture time** ($\max_{\text{cap}}$): This is equal to the single similarity of capture time for the document-event pair ranked first: $sim_{\text{capture}}(d, e_1)$
- **Maximum upload time** ($\max_{\text{upl}}$): This feature equals $\max_{\text{cap}}$ but uses the upload time instead of capture time: $sim_{\text{upload}}(d, e_1)$

This yields a feature vector $\vec{v}_{new}(d)$ for each document d:

$$\vec{v}_{new}(d) = \begin{pmatrix} \max \\ \min \\ \text{avg} \\ \text{stddev} \\ \max_{\text{cap}} \\ \max_{\text{upl}} \end{pmatrix} \tag{5.37}$$

We use that vector to classify d into two classes: belongs to top scoring event (positive) or belongs to a new event (negative). The classifiers are trained using an equal number of positive and negative examples and, as in the above case, return a probability that the document belongs to a new event:

$$P_{\text{new}}(d) = P\left(\text{negative}|\vec{v}_{new}(d)\right) \tag{5.38}$$

Having that probability, we use a threshold as a hyperparameter to be tuned. This threshold decides whether the event is assigned to a new event or assigned to the top-ranked event. The optimal threshold is determined empirically using a gradient descent technique on the training data.

CHAPTER 6

Experimental Setup and Results of the Supervised Single-Pass Classification

After having presented the supervised single-pass classification system in the previous chapter, we now present our experimental setup and the results of the supervised single-pass classification approach. We first start with the description and definition of the measures used evaluation. We then describe individual experiments for single steps of the clustering framework. After that, we present the results of our experiments for the final system. This system is then compared to several baselines.

As our approach consists of multiple steps (candidate retrieval, scoring and ranking, and new event detection), interacting with each other but each tackling a different problem, we develop experimental scenarios for each of these steps individually with the goal to find their optimal setting. These experiments are described in Sections 6.2 to 6.4. The findings of these individual experiments are then taken into account when conducting experiments with the clustering framework system as a whole, further analyzing the interactions between the individual steps.

In general, we use the dataset derived from Flickr which we already presented in detail in Chapter 4 for all our experiments. In each section for the single steps optimization we describe exactly what parts of the dataset were used. All optimization experiments only make use the training set of ReSEED. This is done to avoid that any final results based on the test set are influenced positively or negatively. For the evaluation of the results we use the evaluation methods which we proposed for the dataset (see Section 4.4.3). In the following we start with a definition of the measure used to evaluate the performance of our approach.

6.1 Definition of Evaluation Measures

In Section 4.4.3 we proposed different evaluation measures which are suitable to evaluate clusterings of the ReSEED dataset. The aim of these measures is to allow an easy comparison of our results with the results of others. All proposed measures are common evaluation measures from the text mining and machine learning literature which are typically employed to measure the quality of a clustering. For our experiments, we employ the well-established *F-measure*. Initial experiments with *Normal Mutual Information* showed very high results for many different clusterings which, in our opinion, does not help to judge the real appropriateness of the clustering as well as the comparison with other approaches. Therefore, we only make use of the F-measure.

F-measure is the weighted harmonic mean of precision and recall. There are two different variants: micro- and macro-averaged. In our case, we use the micro-averaged version with a higher computational effort. In this variant, precision and recall are computed and averaged document-wise. In general, precision P and recall R are defined as given in the following two equations:

Definition 6.1.1 — Precision and Recall

$$P = \sum_{d \in D} \frac{1}{|D|} \cdot \frac{|\mathrm{Cluster}(d) \cap \mathrm{GoldStandard}(d)|}{|\mathrm{Cluster}(d)|} \tag{6.1}$$

$$R = \sum_{d \in D} \frac{1}{|D|} \cdot \frac{|\mathrm{Cluster}(d) \cap \mathrm{GoldStandard}(d)|}{|\mathrm{GoldStandard}(d)|} \tag{6.2}$$

The variable Cluster(d) denotes the clustering produced by the classification system to be evaluated and GoldStandard(d) denotes the underlying gold standard to be evaluated against. Having determined both precision and recall, the general F-measure F_β is calculated as follows:

Definition 6.1.2 — General F-Measure

$$F_\beta = \frac{(\beta + 1) \cdot P \cdot R}{(\beta^2 \cdot P) + R} \tag{6.3}$$

The variable β in the equation is a relative weight for precision and recall. In our clustering task it might be desirable to weight the precision as more important. The reason for this is that from the perspective of a user a purer clustering is favored, justifying a higher weight of precision over recall. Nevertheless, for the sake of comparability with other approaches, we refrain from setting β to a different value than 1.

Therefore, we measure the performance of our approach using the F_1-measure defined as follows:

Definition 6.1.3 — F_1-Measure

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{6.4}$$

In the following, all indications of quality information are denoted in terms of F-measure, precision, and recall as defined above.

6.2 Optimizing Candidate Retrieval

In our framework, we propose to use a candidate retrieval step in order to make the overall approach scalable. The main advantage of this step is that we need to compare only those pairs which were proposed by the candidate retrieval approach instead of having to compare the newly arriving document with all events in the current event set. Therefore, we are faced with the problem of finding a minimal number of event candidates among which the correct candidate is included.

In the following, we compare the different candidate retrieval strategies c_i which we proposed in Section 5.3. In the following, we present our method to compare the different candidate retrieval strategies, and conclude with the presentation of our findings.

We follow two strategies to optimize the candidate retrieval step. In this section we present an analysis of the effectiveness of the candidate retrieval itself. Later, we compare the influence of the different candidate retrieval strategies in terms of overall classification performance; this is described in Section 6.5.1.

6.2.1 Experimental Settings

To conduct our experiments, we use the training split of our dataset as a basis. The experiments conducted for the optimization of the candidate retrieval make use of all 306,159 documents included in the training split together with the corresponding gold standard information. The data is not modified in any form.

In preparation for the evaluation, we adapt our framework in a way so that an optimal clustering is simulated. It is our aim to build this clustering simulation approach allowing the clustering to be an optimal iterative clustering process using the gold standard information to assign the documents to their correct events. The new event creation and the assignment to an event is done using the gold standard mapping. This simulation allows the analysis of all existing events at the time of appearance for each document in the dataset and their distance regarding different similarity measures.

In order to analyze the effectiveness of each clustering strategy c_i, we need to determine the number of events to be retrieved still ensuring that the corresponding correct event for the document is included in the event candidate set. To this end, we employ the clustering simulation. For each document in the iterative clustering process, we retrieve the ranks of its corresponding correct event as returned by the different candidate retrieval strategies c_i. This is shown more formally in Algorithm 6.1.

After having determined the individual ranks for each document and candidate retrieval strategy, we are able to count for each rank how often it appeared in the process. Each rank actually

Algorithm 6.1: Determination of ranks for the candidate retrieval strategies using an optimal decision for event creation and assignment

Input : Training stream of ReSEED dataset with documents D and gold clustering E, clustering strategies C

Output : Documents clustered by their event following the gold standard

```
foreach $d \in D$ do
    foreach $c \in C$ do
        $Rank_c(d) \leftarrow$ rank of event to which $d$ belongs or 0 if new event;
    end
    $e_{gold} \leftarrow d(E)$;
    if $d \notin e$ then
        create new event $e'$;
        $e' = \{d\}$;
        $\vec{e'} = \vec{d}$;
    else
        $e_{gold} = e_{gold} \cup \{d\}$;
        recompute $\vec{e}_{gold}$;
    end
end
```

represents the number of needed candidates to be retrieved. If we now want to know for how many documents we can get the correct candidate if a certain number of candidates is retrieved, we simply have to sum up all number of appearances of each rank which is equal or smaller than the number of candidates to be retrieved. Using this technique, we obtain k. This is done for each candidate retrieval strategy c_i individually. Having obtained these values, it is then possible to compute the effectiveness Effectiveness(c_i, k) as proposed in Equation (5.5) for each candidate retrieval strategy c_i and a certain number of candidates k.

6.2.2 Results

In this section we present the results of our optimization experiments regarding the candidate retrieval strategies. We start with an analysis of the effectiveness followed by an examination of the computational costs for each candidate retrieval strategy.

Effectiveness of Candidate Retrieval

We first examine the effectiveness of the single candidate retrieval strategies $c_1, ..., c_6$. These were: retrieval by upload and capture time, by geographic location, and by the three textual features (tags, title, and description). The effectiveness of all these strategies depending on the number k of events retrieved is depicted in Figure 6.1.

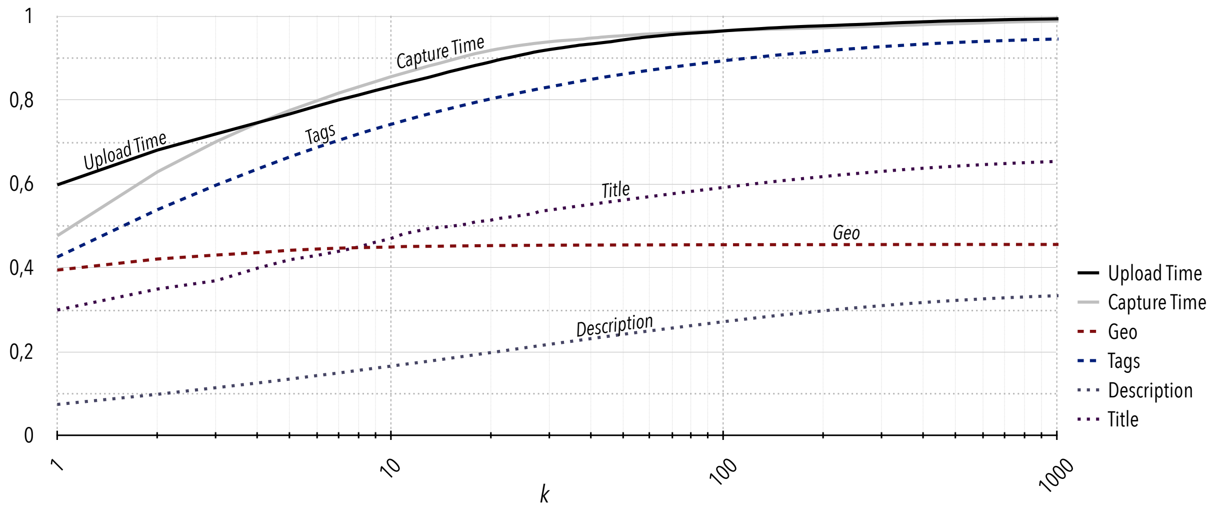


Figure 6.1.: Effectiveness of different (single-strategy) candidate retrieval strategies

We can see that only the candidate retrieval strategies based on temporal features and on TF-IDF values of the tags reach effectiveness scores higher than 80 %, and only the temporal candidate retrieval strategies come close to an effectiveness of 100 %. However, this has the cost of having to retrieve more than 400 to 500 events. It is interesting to see that the geographic location-based strategy reaches a plateau at an effectiveness of about 45 %, but reaches its maximal effectiveness very quickly for $k = 6$. Nevertheless, the effectiveness of the time-based candidate retrieval strategies is always better.

In Table 6.1 we show the number of events that have to be retrieved by each candidate retrieval strategy in order to reach an effectiveness of 30 %, 60 %, 80 %, 90 %, and 95 %, respectively. It is remarkable that only four candidate retrieval strategies reach an effectiveness over 80 %, and only three of the strategies reach more than 90 %, where the number of retrieved candidate events k is still within an acceptable range regarding the scalability for the whole system (with relation to the number of document-event comparisons for each document).

Table 6.1.: Needed number of k to reach x % effectiveness

Candidate Retrieval Strategy	**95 %**	**90 %**	**80 %**	**60 %**	**30 %**
Upload Time	52	21	7	1	1
Capture Time	37	17	6	2	1
Geo	–	–	–	–	1
Tags	906	103	19	3	1
Title	–	–	–	108	1
Description	–	–	–	–	183
Optimal Combination	5	3	2	1	1

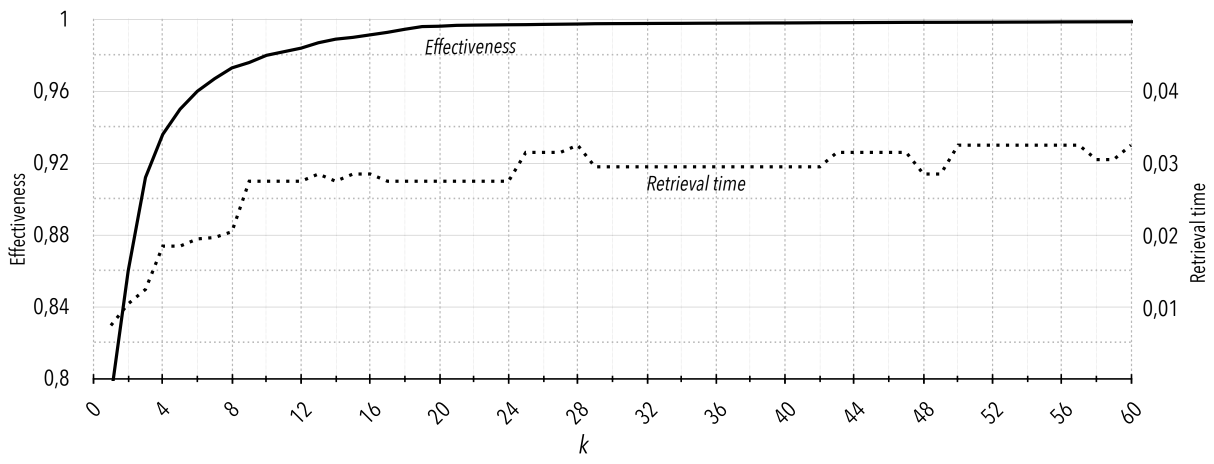


Figure 6.2.: Retrieval time and effectiveness for the optimal combination of single candidate retrieval strategies over k

An important observation regarding the effectiveness is that while the temporal candidate retrieval strategies achieve an effectiveness of 95 % for higher ks (52 and 37), the optimal combination (see Section 5.3.2) reaches this effectiveness after only 6 events retrieved. This is interesting as it suggests that this strategy has a much better cost-effectiveness ratio compared to the timestamp-based strategies.

In Figure 6.2 we plot the effectiveness as well as the retrieval time for the optimal combination strategy over different values of k. This strategy shows that the different strategies c_1 to c_6 together are effective for the retrieval of different events; this means that each candidate retrieval strategy retrieves non-overlapping events. The optimal combination strategy seems to bring an advantages over all single strategies. The combination reaches an effectiveness of 98 % at $k = 10$. Further, the diagram shows that the retrieval time remains fairly constant across different values of retrieved candidates k. However, the retrieval time for any timestamp-based strategy is up to 30 times lower, never exceeding a retrieval time of 0.003 s.

It is important to note, however, that the above observations do not imply that the optimal combination candidate retrieval strategy has the best cost vs. classification performance tradeoff with respect to our classification task. Later, in Section 6.5, we turn to examine the cost-performance ratio of the different strategies with respect to the overall classification performance of the system in terms of F-measure.

Scalability and Processing Time

As can be seen in Figure 6.2, the time needed to retrieve the candidates is fairly constant across different values of k. This is the case for all candidate retrieval strategies. However, the overall processing time required clearly increases with the amount of k. The graph in Figure 6.3 depicts the overall processing time of the system as a whole using upload time-based candidate retrieval over the number k of event candidates considered.

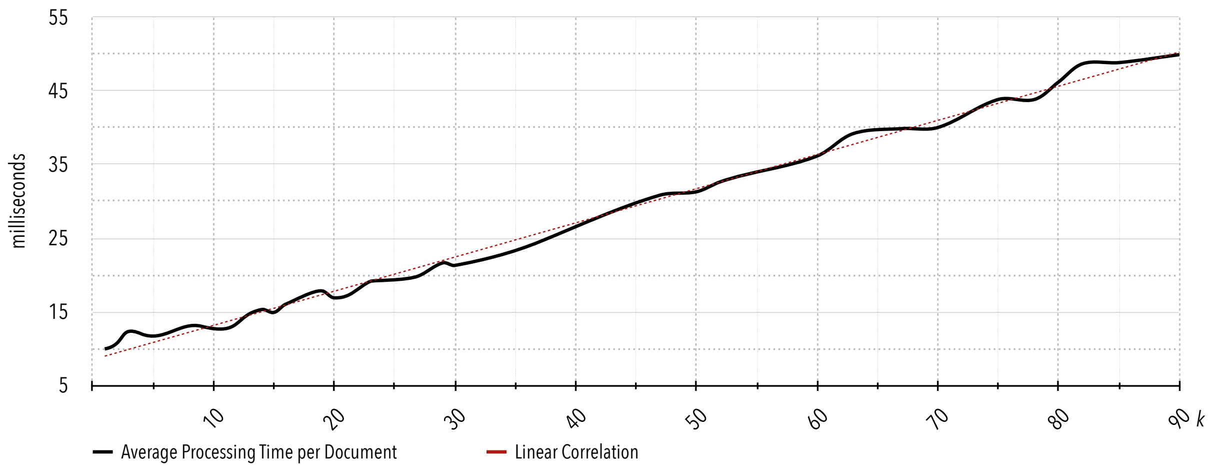


Figure 6.3.: Average processing time for one document over different *k* based on Upload Time candidate retrieval strategy

The retrieval times using the different strategies c_1 to c_6 are different. It is obvious that the lowest retrieval time is achieved using the time-based strategies c_1 and c_2 because a query needing to compare one single integer value can be processed very efficiently using indexed structures in the data storage. The text-based retrieval strategies are the most expensive as many comparisons have to be made. Combining the single strategies as done in the uniform or optimal clustering, the retrieval time is inherently bigger as for each query multiple single queries have to be executed.

We can observe that the computation time per document increases if more candidates are retrieved. This is indeed consistent as more comparisons have to be made. More importantly, the increase in computational cost seems to be linear in k (see also the linear correlation in Figure 6.3). As a consequence, we identify that minimizing k is the key issue towards scaling up the system. Once k is fixed, the computation time for each document is constant regardless of the event database size; this is an important requirement for scalability.

6.2.3 Conclusion

In this section we have been concerned with the question of how to scale up the classification process so that it can be performed in real-time by optimizing the candidate retrieval step. Such a step is crucial to reduce the total number of events that are considered as potential candidates to which the incoming data point could belong to. This enables us to scale up to the data sizes and data rates needed when dealing with social media applications data.

We have experimentally compared different candidate retrieval strategies with respect to their effectiveness as well as computational cost. Using a candidate retrieval strategy, the average processing time for one document has been fixed, thus allowing to process an unlimited number of documents without an exponential growth of needed computation time. Therefore, the introduction of the candidate retrieval step makes it possible for us to cluster a large-scale dataset within a feasible time period. In further experiments we will examine the influence of the candidate retrieval strategies on the overall clustering process and the impact on the final results in terms of quality.

6.3 Learning Similarity Functions

After the extraction of pairwise features for a combination of a document and event, we end up with a feature vector of single similarities. We are faced with the challenge to find an appropriate way on how to combine these single similarities to obtain an overall similarity value for this combination.

We use the single similarity metrics proposed in Section 5.4 as a basis, asking how we can learn an overall similarity function which combines these different measures with the result of a satisfactory measure of overall similarity between a document and an event. Therefore, in this section we experiment with the learning of such similarity functions. We employ different classification algorithms to learn how to combine the single metrics in an optimized way. Having such a metric, we will be able to score document-event pairs. This is the basis to rank document-event pairs. This score is then needed for further classification decisions.

6.3.1 Experimental Settings

In this section we describe our experimental settings to learn similarity functions. First, we describe the parts of the training dataset used for the conduction of the experiments. After that, we describe the strategies for training the classification algorithms. In the last subsection we then explain our leave-one-out strategy which is the basis for the determination of how well a document can be assigned to an event using the decision of the classifier.

Data Splits and Sampling Strategies

We once again use the training split of our dataset for our experiments. This time, the 306,159 documents of the training split are partitioned into seven subsplits containing 43,737 training documents each. Having these seven equal parts, we use them for cross-validation, allowing us to train on one subsplit and to test the model on the six other subsplits.

We use three different strategies to construct the training dataset on the basis of the similarity function learned. These strategies are used with all subsplits. Each data subsplit consists of a subset of all the documents in question. For each document and each centroid representing an event, the corresponding similarity vector $\vec{v}_{sim}(d, e)$ is computed. Pairs $(d_i, e(d_i))$ consisting of a document d_i and the corresponding event $e(d_i)$ are used as positive examples.

Independent of the used sampling strategy we make sure that the training data set is balanced, generating a number of negative examples that matches exactly the number of positive examples. In the following we describe how this is achieved for the three different strategies we consider:

- **Random:** Using the random method, a subset of documents from the training set is retrieved randomly. A negative example is created by computing the similarity vector of the document and a randomly chosen centroid of an event which the document does not belong to.
- **Time-based:** The documents of the events are uploaded to Flickr in a time-ordered manner. Thus, it seems natural to choose n consecutive documents from the training set. We use the first documents appearing in each split. To create the negative examples we use the random strategy as described above.
- **Nearest:** Here, we also choose n consecutive documents from the training dataset as we did in the *time-based* strategy. However, in this case, the negative examples are chosen using the nearest event class where the document does not belong to $neg(d_i^+) = \max_{d_i^- \notin ext(e(d_i^+))} \sum_{i=1}^{6} sim_i(d_i^-, e(d_i^+))$.

Finally, we end up with 21 training sets, one for each of the three strategies for every subsplit.

Training Examples for Classifier Training Strategies

Regarding the creation of the training examples, we consider each training data subsplit individually. We choose m positive and m negative examples from each of these splits according to three sampling strategies described in the previous section.

For m we consider the following values 100, 200, 300, 400, 500, 1000, 2000, 4000, 8000, 16000, and 32000. As classifiers we decided to use a standard SVM, a ranking SVM, as well as a Decision Tree. In order to be able to compare the results of these classifiers we use the same training examples for all three of them.

For the standard SVM and the Decision Tree we use the class labels "1" for positive and "-1" for negative examples. For the ranking SVM we have to group positive and negative examples into a ranked group where the document-event pair $(d_i, e(d_i))$ gets a higher rank (label "2") than the pair (d_i, e') where the same document is compared to a centroid of an event cluster it does not belong to (label "1").

Classifier Settings

In our experiments we rely on three different classifiers. Therefore, in the following we describe the settings and parameters which we have chosen for the different classifiers to be tested. Additionally, we mention which implementation is used. We feed all classifiers with the same training data for equal scenarios; both types of SVMs and the Decision Tree are trained using the same positive and negative examples.

- **Support Vector Machine**: For the standard SVM we make use of the *LibSVM* implementation from Chang and Lin [CL11]. Ultimately, we use the C-SVC implementation for support vector classification. The trade-off between training error and margin C is set to 1.0. In order to yield an actual similarity measure, the output of the SVM has to be normalized into values in the interval [0..1]. For this purpose LibSVM provides probability estimates on which we rely. These probability estimates are derived from the distance to the decision boundary. For our experiments we experiment with two different kernels:

Definition 6.3.1 — Linear and Radial Basis Function (RBF) kernel

$$\Phi(d, e) = \langle d, e \rangle \quad \text{(Linear kernel)} \tag{6.5}$$

$$\Phi(d, e) = \exp(-\gamma \cdot ||d - e||^2), \quad \gamma = \frac{1}{6} \quad \text{(RBF kernel)} \tag{6.6}$$

- **Ranking Support Vector Machine**: As ranking SVM we use SVM^{rank}, an implementation from Joachims [Joa02]. Instead of the standard C value of 0.01 we use a training error and margin trade-off of 1.0 for comparability with the standard SVM. With this classifier, we also use the above mentioned kernels.
- **Decision Tree**: For the Decision Tree we use the CART (Classification and Regression Trees) implementation (see Breiman et al. [Bre+84]). The binary tree is built using the feature and threshold which yield the maximal information gain at each node. We set the tree to have a maximum depth of 5, the minimal samples per leaf is set to 1. The CART implementation also returns normalized values in the interval [0..1].

Leave-One-Out Strategy

For conducting the experiments we use a leave-one-out strategy. In particular, we use our test set (a subsplit of the training set) and choose k event clusters. We then take exactly one document from these event clusters. The centroids of the event clusters are then recomputed as if the document taken out was never part of it. This is done for each data split individually.

Actually, 1,000 event clusters ($k = 1000$) from each data split are selected and a random document from each of them is taken out. We finally end up with 1,000 clusters and 1,000 documents. In our experiments we then compare every document to every centroid of the event clusters. As we know the correct assignment for each document, it is possible to measure the correct assignment rate for each method. This accuracy is averaged over all seven train-test split pairs.

6.3.2 Results

In the following we present the results of the experiments regarding the similarity function learning. In Table 6.2 we show the average accuracy for the standard Support Vector Machine with the two different kernels together with the different sampling strategies and a different number of training examples. Table 6.3 shows the average accuracy for the ranking Support Vector Machine with the same settings as used for the standard SVM. The results for the

Table 6.2.: Average assignment rate using a standard SVM

Training examples	Linear Kernel Time-based	Linear Kernel Random	Linear Kernel Nearest	RBF Kernel Time-based	RBF Kernel Random	RBF Kernel Nearest
200	98.0 %	97.3 %	97.7 %	98.1 %	97.8 %	97.7 %
400	98.9 %	97.5 %	98.7 %	98.6 %	97.6 %	98.8 %
600	98.8 %	97.4 %	98.5 %	98.4 %	97.5 %	98.9 %
800	98.5 %	97.4 %	98.4 %	98.4 %	97.7 %	98.7 %
1000	98.6 %	97.7 %	98.4 %	98.6 %	98.0 %	98.6 %
2000	98.3 %	97.9 %	97.7 %	98.4 %	98.0 %	98.9 %
4000	98.4 %	98.0 %	98.8 %	98.3 %	98.3 %	**99.1 %**
8000	98.4 %	98.1 %	98.9 %	98.5 %	98.4 %	**99.1 %**
16000	98.6 %	98.2 %	**99.1 %**	98.7 %	98.5 %	98.4 %
32000	98.7 %	98.4 %	**99.1 %**	98.7 %	98.6 %	98.1 %
64000	98.7 %	98.5 %	99.0 %	98.9 %	98.7 %	96.9 %

Table 6.3.: Average assignment rate using a ranking SVM

Training examples	Linear Kernel Time-based	Linear Kernel Random	Linear Kernel Nearest	RBF Kernel Time-based	RBF Kernel Random	RBF Kernel Nearest
200	98.6 %	98.4 %	98.9 %	98.2 %	97.8 %	97.6 %
400	98.9 %	98.4 %	98.9 %	98.4 %	98.4 %	98.1 %
600	98.9 %	98.5 %	98.9 %	98.4 %	98.3 %	98.2 %
800	98.8 %	98.4 %	98.8 %	98.4 %	98.2 %	98.2 %
1000	98.8 %	98.4 %	98.8 %	98.4 %	98.3 %	98.1 %
2000	98.9 %	98.5 %	99.0 %	98.5 %	98.4 %	99.0 %
4000	98.9 %	98.5 %	99.0 %	98.5 %	98.7 %	99.0 %
8000	98.6 %	98.5 %	98.9 %	98.5 %	98.2 %	**99.1 %**
16000	98.6 %	98.5 %	99.0 %	98.5 %	98.2 %	99.0 %
32000	98.6 %	98.5 %	99.0 %	98.5 %	98.3 %	99.0 %
64000	98.7 %	98.5 %	99.0 %	98.5 %	98.5 %	98.9 %

Decision Tree are shown in Table 6.4. The results are average over six subsplits not including the subsplit used for the training of the classifier.

Overall, the problem is solvable using all three classifiers. However, we see that the performance varies. Both the parameters and the type of classifier chosen have a huge impact on the accuracy of the decisions.

Impact of Classifier and Kernel Influence

The results allow the conclusion that a Support Vector Machine suits better for solving the problem. Even though the Decision Tree reaches over 95 % of accuracy when trained with

Table 6.4.: Average assignment rate using a Decision Tree

Training examples	Time-based	Random	Nearest
200	0.5 %	4.0 %	22.2 %
400	6.8 %	1.2 %	27.2 %
600	0.1 %	4.6 %	34.5 %
800	3.5 %	3.3 %	42.5 %
1000	0.2 %	4.0 %	33.8 %
2000	3.1 %	5.5 %	55.9 %
4000	3.9 %	11.5 %	63.8 %
8000	13.1 %	22.9 %	90.2 %
16000	37.4 %	33.0 %	93.2 %
32000	42.0 %	40.3 %	94.7 %
64000	59.9 %	54.3 %	**95.8 %**

64,000 examples, both types of SVMs easily outperform the Decision Tree reaching over 99 % accuracy.

One very surprising conclusion of the experiments is the fact that a good similarity function can already be learned with very few examples when using a Support Vector Machine. Using a ranking SVM, it is possible to already learn a model with nearly 98 % accuracy with only two training examples—one positive and one negative. In contrast to that, a standard SVM needs already more than 20 examples (10 positive and 10 negative) to learn a model with around 97 % accuracy.

While the performance of the ranking SVM does not vary much using different amounts of training data, the standard SVM clearly suffers from overfitting; this is shown by the drop of performance from around 16,000 training examples onwards. The ranking SVM does not suffer from such a performance drop, thus showing a more robust behavior.

There is not too much difference in the maximal performance regarding the kernel used. Both types of SVMs reach 99.0 % to 99.1 % accuracy with either kernel, leading to the assumption that both kernels can be used to tackle the problem. The usage of a linear kernel has advantages over the RBF kernel regarding the computational time needed. As this has an impact on the overall scalability of our system, the standard SVM using a linear kernel is the preferred method to learn a similarity function.

Impact of Sampling

It can be easily observed from the results that the sampling strategy and thus the choice of how many training examples are used for a certain classifier is crucial for the creation of a model as the quality varies a lot. It is also notable that a Support Vector Machine needs a lot less training examples than a Decision Tree to create a good classification model.

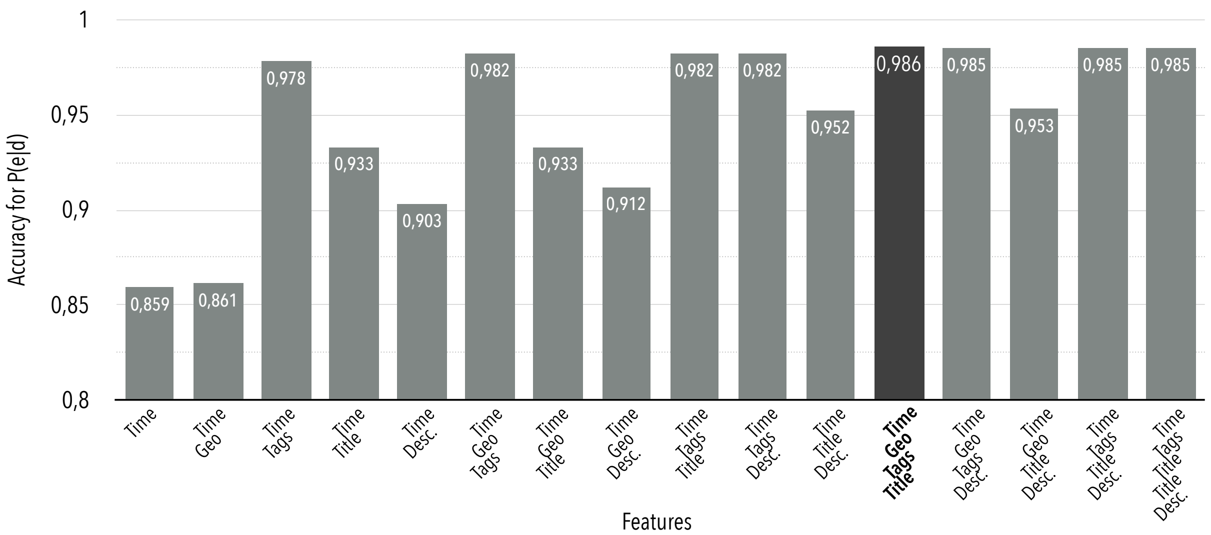


Figure 6.4.: Feature analysis for computation of P(e|d) of accuracy using a standard SVM with linear kernel

We see that using the *random* sampling strategy leads to the worst results. It may be possible that a model created by a random selection of positive and negative examples gives acceptable results but in general the results are worse and more unstable than using other methods. Overall, more training examples are needed using this sampling strategy to get a reasonable accuracy rate.

Both of the other sampling strategies work very well showing clearly that the search for the nearest wrong pair helps to create a better model. Using the Support Vector Machines we observe this effect using the time-based sampling strategy, too. This is not the case for the Decision Tree which is the classifier most sensitive to the sampling strategy.

Feature Analysis

In Figure 6.4 we compare the accuracy rates used to calculate $P(e|d)$ over the time features and all possible combinations of other features together with the time features. The two time features are always used because this type of data is available for each data point in the dataset. To obtain these figures, we averaged over all accuracy rates for $m \in \{1000, 2000, 4000, 8000, 16000\}$ using the nearest sampling strategy. As classifier we employed the standard SVM with a linear kernel.

Usually, we expect the accuracy to be very low when using the time features only, but in our experiments the use of these features yields an accuracy of about 85 %. We have two explanations for this effect: i) the number of overlapping events regarding their time range is not very large and ii) we do not use one single time feature but both timestamps together (upload an capture time) so that a better classification is possible. In earlier work we showed that using only one single feature results in very poor accuracy rates (see Reuter and Cimiano [RC11]). Regarding the first reason, it nevertheless seems possible that the dataset is still sparse enough to achieve a decent accuracy for the time feature. Even if this is the case, we are aware that if (a lot) more data points are used, the results of a pure time-based clustering will be very poor as we have shown in our work from 2011. Therefore, the usage of the timestamps as a single feature is eventually not as good as implied by the numbers in the figure. Fortunately, about 99.9 % of all documents in the dataset have at least one more feature associated in addition to the time features, therefore allowing for a better decision.

It is remarkable that the additional use of the geographic location does not improve the results much compared to the accuracy using timestamps only. We explain this with the enormous lack of location information in the dataset. Therefore, the decision often depends only on the timestamp-based comparison as no other feature is available; this explains the rather low score. Luckily, more recent camera devices support the geographic tagging by GPS resulting in more pictures including the geographic feature.

Mostly, the addition of a feature does not worsen the accuracy score. The only exception is the usage of all features instead of the best combination (time, geographic information, tags, and title). As the difference is only slight and could change if another data basis is used, we do not see the need to introduce exceptions where features are excluded depending on the availability of other features. Therefore, we always use as many features as are present for the document-event pair.

It is interesting that all combinations of the time features and at least one other feature are enough to produce reasonable results. As 99.9 % of all documents have at least one feature available in addition to the time features, we can conclude that the availability of all features is not compulsory and the lack of features can be compensated for by other features. The results license also the conclusion that the tag feature is one of the best features available as any combination with the tag feature produces very good results. This is especially good as this feature is available for more than 95 % of all documents.

6.3.3 Conclusion

In this section we addressed the question of how we can learn a suitable similarity function from training data. We used different classifiers, a Decision Tree as well as two different types of Support Vector Machines (ranking and standard) in order to train an appropriate similarity function. Furthermore, we investigated the impact of the amount of training data and different strategies for creating training data. As a result we have seen that it is possible to learn a suitable similarity measure using all of these classifiers. When using Support Vector Machines it is already sufficient to use only few examples to train a model.

We have shown that the use of well-chosen training examples improves the quality of the assignment significantly. The sampling strategy is crucial for the success maximization of the training process. It has been shown that the search for the nearest wrong document-event pair helps creating a better decision model. In general, we need only few data points to be able to create a good decision model. However, if too much data is used for training, at least the standard SVM suffers from overfitting.

In addition to the results above, our feature analysis leads to the conclusion that the lack of a single feature has no great impact on the quality of the prediction. The accuracy rates are still acceptable even if more than one feature is missing. As nearly 99.9 % of all documents have more than the time features assigned, the methods we proposed in this section are applicable in a real-world scenario.

The results presented in this section are comparable to those from earlier work (see Reuter and Cimiano [RC11]). The experiments were conducted using a different dataset and the setting differed from the one used in this section. However, we came to the same conclusion that the problem can be solved best using a standard SVM. We propose to use this classifier with a linear kernel.

6.4 New Event Detection

After the successful scoring and ranking of the document-event pairs, we have to answer the question whether the newly arrived document belongs to an existing event or if we have to create a new event. To this end we have to decide if the newly arrived document is similar enough to one of the events seen so far. From the scoring and ranking step we can deduce several features (see also Section 5.6) to decide on this question. These features can be used to train a classifier which then can be employed to predict whether a new event should be created or not.

6.4.1 Experimental Settings

In the following we describe the experimental setup to learn a classifier for the detection of new events. As described in Section 5.6 we use the feature vector $\vec{v}_{new}(d)$ with derived features from the preceding step of the framework to learn a model for the prediction whether a new document belongs to an already existing event or should be classified as a new event.

For the task of new event detection, we have to investigate which combination of features in the feature vector $\vec{v}_{new}(d)$ should be considered in order to maximize the accuracy of the decision.

Data Creation for Classifier Training

For the training process of the new event classifier, we also use the training split of our dataset. We make use of the seven subsplits with 43,737 documents each as proposed before. As the number of positive examples for this classifier is limited by the number of real events in each subsplit (around 2,000 events) following the gold standard, we have to merge some subsplits in order to be able to feed the classifier with more than 2,000 training examples. Therefore, we use subsplit 1 to 4 and merge these subsplits resulting in a split of 174,948 documents which are spread over 9,046 events. This data is used for the training of the the classifier. The optimization is then made on the merged subsplits 5, 6, and 7 resulting in 131,211 document spread over 6,536 events.

In order to obtain the new event feature vector $\vec{v}_{new}(d)$, we employ an optimal clustering simulation similar to the simulation proposed above. The calculation of the similarities between the document-event pairs in the scoring and ranking step is done using the learned similarity metric from the previous section. For this step we employ a standard SVM with a linear kernel trained with 8,000 positive and negative examples as proposed. Nevertheless, the documents are placed into their corresponding correct event cluster according to the gold standard independent from the results of the classifier in the scoring and ranking step.

As we later want to use a candidate retrieval strategy in our final optimized clustering framework, we also have to take this into account at this step. This is important as the calculation of the average and standard deviation for the new event feature vector as well as the determination of the minimum value heavily depends on this. Therefore, we use the clustering simulation together with the candidate retrieval strategy retrieving the 100 nearest events based on their upload timestamps.

Finally, the optimal clustering is executed using the merged subsplits 1 to 4. In each step of that clustering process we calculate the similarities between the document-event pairs. In addition to that we know if a new cluster has to be created or not, this gives us the possibility to know whether the new event feature vector created is positive or negative. As a result of our optimal cluster simulation, we get 9,045 positive (is a new event) and 165,902 negative (is not a new event) examples for the feature vector $\vec{v}_{new}(d)$ including all the feature described in Section 5.6.

We use the same clustering simulation with the 131,211 documents from subsplits 5, 6, and 7 to obtain data for the evaluation of the trained new event detection classifier. This process yields 6,535 positive and 124,675 negative examples.

Classifier Training Strategy

The data in the feature vector $\vec{v}_{new}(d)$ is not ordered. Therefore, a nearest sampling strategy as presented in the previous section is not possible to use as it is not clear how distant two data points are. As a consequence, our strategy is to use the first m positive examples appearing together with m randomly chosen negative examples. For m we consider the values $2, 10, 50, 100, 200, 300, 400, 500, 1000, 2000, 4000$, and 8000.

As it has been shown that it is important that the training data set is balanced, we also make sure that the number of negative and positive examples exactly match during the classifier training. To train the classifier, we also use the class labels “1” for positive (is a new event) and “-1” for negative (is not a new event) examples. The training of the Support Vector Machine is done only using the merged subsplits 1 to 4.

Classifier Settings

For our experiments we have chosen a standard Support Vector Machine as classifier for the new event detection. As for the SVM used in the scoring and ranking step, we again use the *LibSVM* implementation from Chang and Lin [CL11]. Furthermore, we use the C-SVC implementation for support vector classification. We set the trade-off between training error and margin C to 1.0. The classifier is used with a Radial Basis Function (RBF) kernel.

Even though a binary decision from the SVM is sufficient to decide whether a new event is created or not, we prefer the output of the SVM to be normalized into values in the interval [0..1]. We do this in order to further optimize the decision by empirically determining a suitable threshold. For this, we use the probability estimates provided by LibSVM.

Evaluation Strategy

Our goal is to test how well the classifier predicts whether a new event should be created or not. Therefore, for the evaluation of the classifier models, we use the data created by the optimal clustering process on the merged subsplits 5, 6, and 7. As we know the correct assignment for each data point in this test set, we can measure the correct assignment rate. The resulting accuracy scores can then be used to determine the best parameters for the new event detection classifier.

6.4.2 Results

In this section, we present our results of the optimization process of the new event detection classifier starting with an analysis of the influence of the number of training examples. We then present the results of our feature analysis before we end with a statement whether to use a classifier ore a simple threshold for the detection of new events.

Influence of Training Examples

In our first experiment we take a look at the influence of the number of training examples used to train the classifier. We aim to answer the question how many training examples are needed in order to achieve the best assignment rate using all features of the new event feature vector. To conduct the experiment, the Support Vector Machine has been trained with m positive and m negative examples. The resulting models are then used to predict for each test vector $\vec{v}_{new}(d)$ included in the test set whether it constitutes a new event or not. The results are shown in Table 6.5.

We can observe that the models created using 4,000 or 8,000 training examples give the best results regarding accuracy. It is interesting that already few examples may be sufficient to create a working model. However, if only few examples are used, the quality depends massively on the examples used. Beginning with 100 examples, the quality starts to be stable.

Feature Analysis

To make sure we identify the optimal features to be used for the new event decision, we conduct a feature analysis. The aim of this analysis is to find the optimal combination of features to be used for a decision model. To this end we consider the accuracy of the decision of the model with different combinations. We set the number of training examples m for the SVM classifier

Table 6.5.: Accuracy of new event detection for different number of training examples

Training examples (pos. + neg.)	Accuracy
4	75.2 %
20	68.4 %
100	82.5 %
200	82.8 %
400	84.8 %
600	85.0 %
800	84.8 %
1000	84.4 %
2000	85.2 %
4000	**85.7 %**
8000	**85.7 %**
16000	84.6 %

to 2,000, i. e. the standard Support Vector Machine used is fed with 2,000 positive and negative examples. This setting has been shown effective for creating a well-adapted decision model for solving the new event decision problem.

For our investigation we conducted a greedy strategy to find the best feature combination. That is, we add the next best feature to the feature set in each step as long as the accuracy of that combination is increased. For the test, we use the six features described in Section 5.6. The results of the greedy search are shown in Figure 6.5.

The graphic in Figure 6.5 depicts the steps of this greedy search strategy. It seems natural that the best single feature is *max* which is confirmed by the feature analysis. Nevertheless, we have shown that a combination of four features yields the highest value with about 86.2 % accuracy. These four features are: *max*, *stddev*, max_{upl}, and *avg*. We identified these features already in previous work (see Reuter and Cimiano [RC12]) with the only difference that the order of the feature was slightly different.

Classifier vs. Threshold

For the decision whether to create a new event or not, there is an alternative to using a classifier. It is possible to use a simple threshold on a single feature. A predestined feature for this is the usage of the *max* feature, the similarity of best ranked document-event combination. If this similarity is higher than a certain threshold, a new event is created. Otherwise the document is assigned to the top-ranked event.

This methodology has been proposed by others (e. g. Becker, Naaman, and Gravano [BNG10]). Actually, the use of a simple threshold is equal to the use of a classifier with only one feature. It differs in the process of how the threshold is learned during the training process. We have seen in our feature analysis that the single feature *max* can be used for the new event decision.

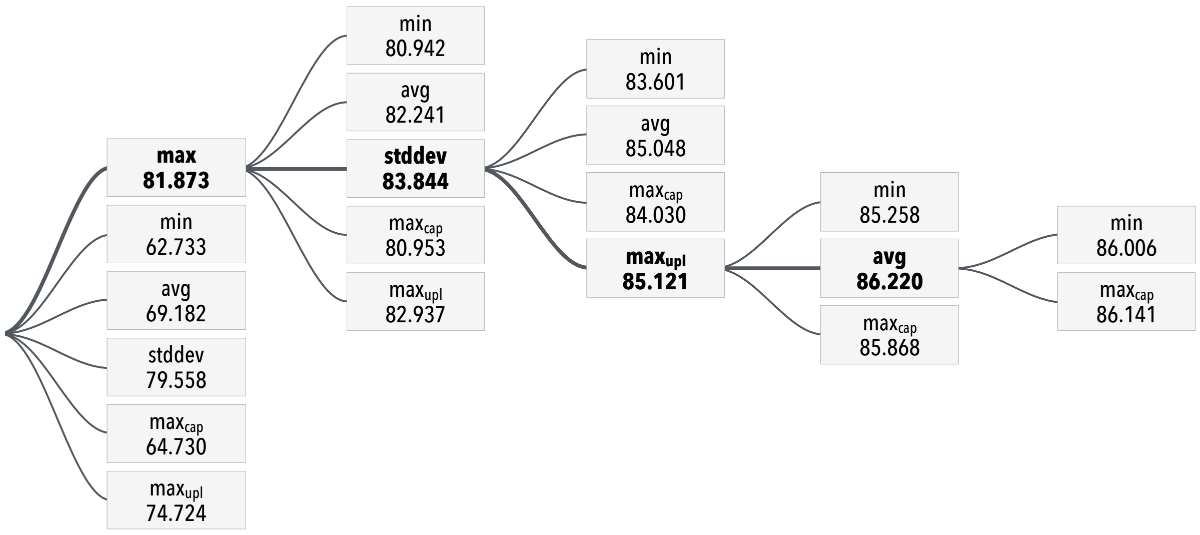


Figure 6.5.: Greedy search for optimal features for the new event detection task using a standard SVM

However, the accuracy rate achieved is not as high as the best combination found (81.9 % vs. 86.2 %). Therefore, the usage of the feature combination together with a classifier yields a higher precision at the cost of some additional computational time.

Our conclusion is thus that the usage of a classifier is the preferred method. The experiments show clearly that the results are inferior when using a simple threshold.

6.4.3 Conclusion

In this section we have been concerned with the question of how we can decide efficiently whether a newly incoming document constitutes a new event or not. We showed that the usage of a classifier with multiple features is preferred over an approach using a simple threshold on one feature for the new event decision.

In order to train an SVM, the number of training examples set to 2,000 positive and the same number of negative examples fits best for a classification model maximizing the decision performance. We also found out that a combination of four features, *max*, *stddev*, max_{upl}, and *avg*, yields the best performance with an accuracy rate of 86.2 %. These features were selected from a set of six features. These features were originally deduced as a result from an optimal clustering.

To conclude, we have presented a working model for the new event detection decision using a SVM as classifier reaching 86.2 % accuracy. We prefer this model over a simple threshold model reaching only 81.9 % accuracy.

6.5 Framework as a Whole – Results and Comparison

Having examined all relevant parts of the clustering framework in detail, we now take a look at our clustering framework as a whole and analyze its overall performance. In Section 6.5.1, we start with a description of how the system parts are optimized in order to allow for a high performance on the overall task. Then, we outline the baselines to which we compare our approach. We conclude this section with the presentation of the results.

6.5.1 Training and Optimization of the System Parts

In the previous sections we took a look at the single steps of the framework and we evaluated how they could be optimized best. While the results indeed optimize each single task, it does not necessarily mean that the optimized single steps of the framework result in a system reaching the best overall performance. Therefore, we now investigate which settings have to be chosen for the single steps in order to achieve on optimal overall clustering.

We start to take a look at the candidate retrieval step again in order to determine an optimal candidate retrieval for the overall task. After that, we explain how we trained the classifiers of the scoring and ranking as well as the new event detection steps for our final clustering framework.

Candidate Retrieval

In Section 6.2.1 we presented how the candidate retrieval strategies perform in terms of effectiveness. We now evaluate how we can improve candidate retrieval further.

We have shown that an optimal combination of different single candidate retrieval strategies maximizes the performance regarding the cost-effectiveness ratio. However, for practical reasons, it is not possible to use the optimal combination in the final system because this strategy requires additional knowledge which is not present in the system in a real-world setting. In particular, we do not know which single strategy has to be used in order to retrieve only a minimal number of events as candidates.

Therefore, a solution for the final approach is using the *uniform* candidate retrieval strategy where an equal number of candidates from each single strategy has to be retrieved. As a consequence, not only the number of candidates retrieved is a lot higher but also the costs for the retrieval time are much higher. This increase of candidates also increases the overall processing time for one document, making the uniform strategy more unattractive in comparison to the optimal strategy.

Besides the two combination candidate retrieval strategies, we saw that only those candidate retrieval strategies that incorporate timestamp features are capable of reaching an effectiveness of almost 100 %. Other strategies do not reach a high enough effectiveness to be taken into account for further examination.

Given the importance of having a suitable candidate retrieval strategy that is tailored to the domain in question, in this section we investigate different candidate retrieval or blocking strategies in terms of their cost-effectiveness tradeoff with respect to the task of clustering a stream of social media documents into an evolving set of events. We show that using a blocking strategy fulfills all criteria demanded above.

In the following we thus compare the most effective candidate retrieval strategies and study their influence on the overall task of classifying documents into their corresponding event with respect to their precision, recall, and F-measure. In addition to that, as a baseline, we compare how well the clustering process performs if no candidate retrieval is applied at all.

Classification Performance

In Figure 6.6 we have plotted the F-measure for the four most effective candidate retrieval strategies: *Capture Time*, *Upload Time*, *Uniform Combination*, *Optimal Combination*. In summary, the results are surprising. It is remarkable that the time-based candidate retrieval strategies outperform both the uniform and optimal combination in terms of F-measure.

The *Upload Time* strategy performs best. We can see that this strategy already yields F-measure values greater than 80 % for a relatively low k. From the view of computational time needed this is pleasing as the retrieval cost for this strategy is also the lowest of all strategies presented as there is only one feature involved. In addition, the retrieval can be implemented by a single query to a database or data storage.

Figure 6.6 also shows the precision and recall values over k for the mentioned strategies. In general, the precision decreases while the recall increases with a growing k; this is as expected. It is observable that the highest recall is achieved by the *Upload Time* strategy, which is to be preferred if recall is important, i. e. if all documents which actually belong to one event should be grouped together by the classifier. The *Capture Time* strategy, however, has the highest precision compared to all other candidate retrieval strategies. Thus this is indeed the preferable strategy if precision is important.

To tackle the question how the system performance is influenced by the candidate retrieval strategy, we compare F-measure, precision, and recall values for different strategies and $k = 18$ (most promising value) with a configuration of our clustering system that uses no candidate retrieval strategy at all. The results are shown in Table 6.6.

Table 6.6.: Comparison of several candidate retrieval strategies with $k = 18$ to no candidate retrieval

Strategy	F_1-Measure	Precision	Recall
Upload Time	**0.876**	0.937	0.823
Capture Time	0.864	0.926	0.810
Uniform Combination	0.849	0.829	0.839
Optimal Combination	0.856	0.847	0.851
No Candidate Retrieval	0.671	0.572	0.813

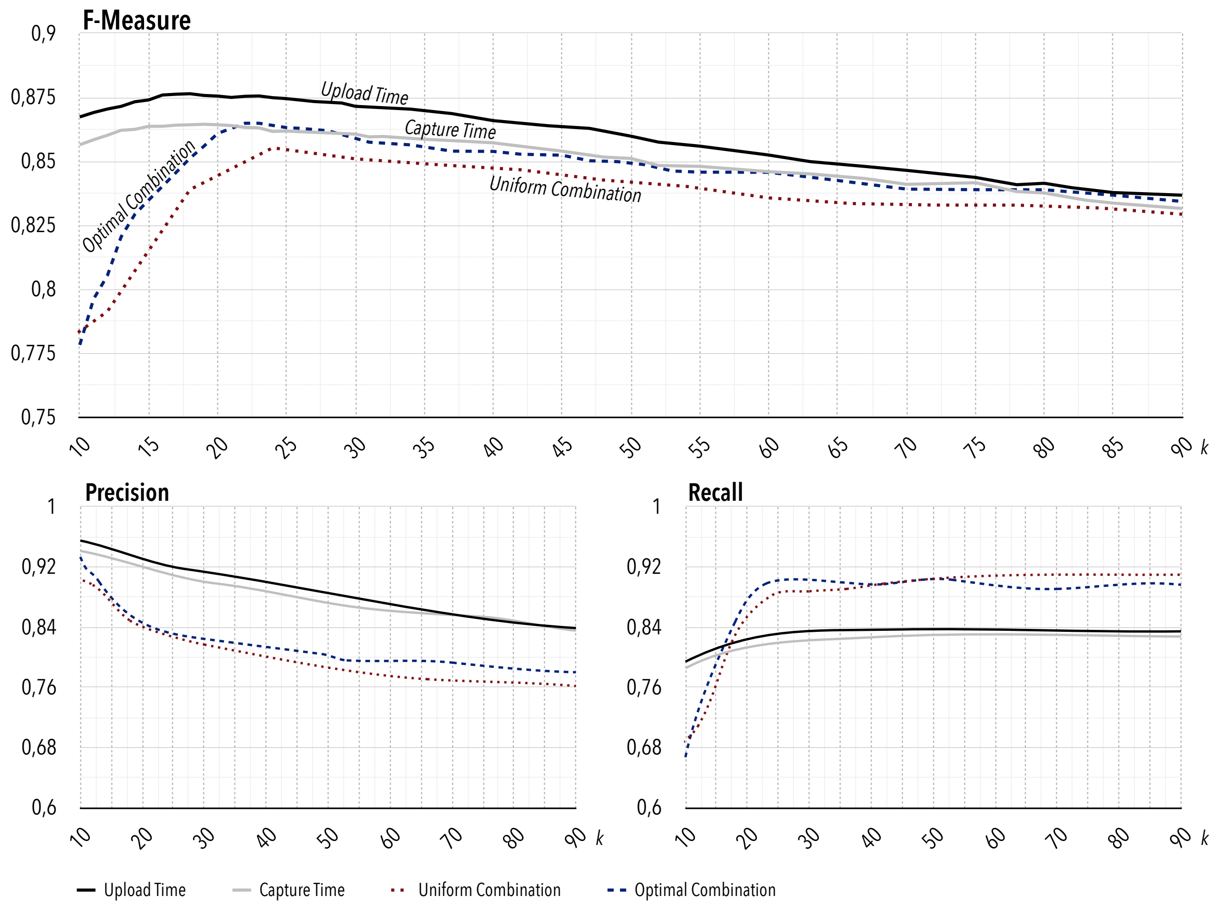


Figure 6.6.: Performance comparison for different candidate retrieval strategies over k[1]

The results show that the *Upload Time* strategy provides the best results and outperforms all other forms of candidate retrieval for $k = 18$ and with respect to F-measure. Another surprising finding is that, in spite of considering only 18 events, all candidate retrieval strategies outperform a configuration of our system where no candidate retrieval strategy is used at all. The reason for this is that the candidate retrieval step eliminates noise that might confuse the classification components.

Processing Time

In Figure 6.7 we plot the overall processing time for a full classification process of our framework. In that graphic we compare the overall processing time for one document after a certain number of documents processed. We see that the processing time for one document increases constantly if no candidate retrieval strategy is used. Using such a strategy, the processing time is almost

[1]Values for the uniform strategy are interpolated for all values not dividable by 6 as by definition, this strategy allows only an increase of six candidates at once.

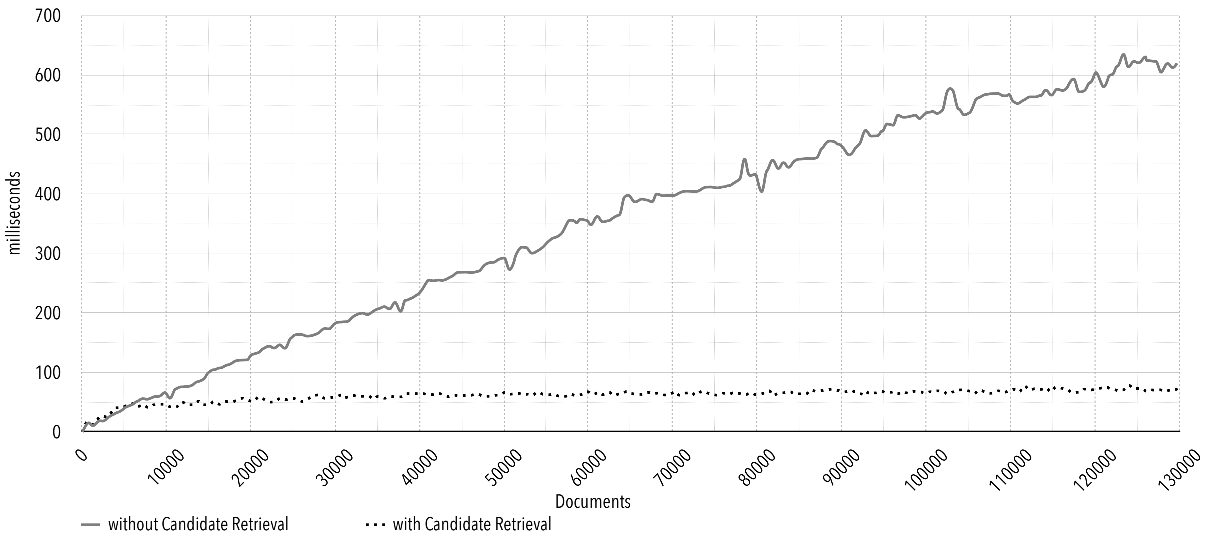


Figure 6.7.: Computing time for document processing using candidate retrieval in comparison to no candidate retrieval

constant over time. The small increase over time can be explained by longer retrieval times of the candidates as the event set grows.

Summary

Overall, our findings and results allow for the following conclusions:

- Our candidate retrieval strategies clearly outperform an approach not using any of those strategies at all.
- Candidate retrieval strategies with the highest cost-effectiveness ratio (e. g. *Optimal Combination*, as presented in Section 6.2) are not the best ones with respect to the overall classification performance. They are outperformed by simpler strategies that consider only the timestamp (*Capture Time* and *Upload Time*), which moreover can be computed efficiently by a single query, not requiring to combine results from multiple queries.
- Our system reaches its top performance of 87.6 % F-measure using the *Upload Time* candidate retrieval strategy with only 18 events retrieved.

These results are very interesting as they have significant impact on the design of any system attempting to classify social media documents into an evolving set of events.

Scoring and Ranking Step

Regarding the decision for assignment of data items to events we use the created training data described in Section 6.3. This is a balanced training set with 8,000 positive and 8,000 negative samples from the training split of our dataset which have been created using the *nearest* sampling strategy.

The training data is used to train a standard Support Vector Machine (C-SVM) together with a linear kernel. The hyper-parameter C denoting the trade-off between the training error and margin is set to 1.0.

In difference to the classifier used for optimization, the SVM classifier here has been extended so that it can handle missing features instead of assuming that the similarity for a missing feature (e. g. the geographic location) is 0.0. This enables the classifier to distinguish between the case where the feature is missing and the case where the similarity is actually 0.0.

New Event Detection Step

We created 2,000 positive and 2,000 negative training examples as proposed in Section 6.4, using the trained scoring and ranking classifier and the candidate retrieval method as described above.

The SVM is trained using the same settings as the one for the scoring and ranking step with the difference that we use an RBF kernel of the form $\Phi(d, e) = e^{-\gamma \cdot ||d-e||^2}$ with $\gamma = \frac{1}{4}$.

We also optimized the value for the hyper-parameter θ_n, which specifies whether the newly arrived data point belongs to a new event or not. It has been determined by gradient descent on the training set of ReSEED using F-measure as optimization criterion. We get an optimal value of 0.63 for our approach, and a value of 0.48 for our re-implementation of the approach by Becker, Naaman, and Gravano [BNG10].

6.5.2 Baselines

As baselines for compsarison we use the following ones:

- **PerDay**: Simple day-wise clustering (upload and capture time)
- **PerUserDay**: Clustering by day (upload and capture time) and uploader
- **Becker**: CLASS-SVM method described in Becker, Naaman, and Gravano [BNG10]
- **BeckerCR**: CLASS-SVM method with candidate retrieval

Using the simple day-wise clustering *PerDay*, we take the test split of our dataset and merge all documents into one cluster day-wise. That is, all documents from one single day between 0:00:00 and 23:59:59 are clustered into one event. We do this both for the upload and capture timestamp individually.

The baseline *PerUserDay* where the documents are merged by day and uploader is similar to the previous one. It is different insofar that the uploader of the document is also taken into account. Ergo, all documents of one uploader from one single day between 0:00:00 and 23:59:59 are put into one single event. The idea behind this baseline is the imitation of many single-user

Table 6.7.: Results of our approach in comparison with different baselines using the ReSEED test set

Method	F_1-measure	Precision	Recall
Our Method	**0.886**	0.951	0.830
Becker	0.660	0.558	0.806
BeckerCR	0.882	0.909	**0.856**
$\text{PerDay}_{\text{cap}}$	0.596	0.491	0.758
$\text{PerDay}_{\text{upl}}$	0.607	0.494	0.786
$\text{PerUserDay}_{\text{cap}}$	0.841	0.980	0.736
$\text{PerUserDay}_{\text{upl}}$	0.842	**0.996**	0.729

photo album systems (like Apple iPhoto or Adobe Lightroom) where photos are clustered into events day-wise. This method is also applied using both timestamp types individually.

Becker, Naaman, and Gravano [BNG10] describe an incremental clustering approach which also makes use of a Support Vector Machine in order to find the most likely event that a document belongs to. There are two crucial differences to our approach. First, they do not use a second decision model to detect new events, but a simple threshold based on the maximal similarity. Second, they do not employ a candidate retrieval step; therefore, Becker, Naaman, and Gravano [BNG10] have to scan all events in the database for each incoming document. We have reimplemented their approach and tested it under the same conditions and on the same dataset as our approach.

As another baseline, we use the re-implementation from the approach of Becker et al. and add a candidate retrieval step. With this enhancement their approach can thus scale to larger datasets. It is also tested under the same conditions as our approach.

6.5.3 Overall System Performance

In the following, we compare the overall system performance of our framework in terms of precision, recall, and F-measure to the performance of the baselines. In particular, the following systems are compared: *Becker*, *BeckerCR*, $PerDay_{cap}$, $PerDay_{upl}$, $PerUserDay_{cap}$, $PerUserDay_{upl}$, as well as our method.

The results of the different approaches are reported in Table 6.7. All experiments have been conducted on our ReSEED test set and can thus be compared to other approaches using this dataset for clustering. The key observations are as follows:

- The *PerUserDay* baseline methods reach decent F-measures. Furthermore, the precision is very high, clearly showing that the simple assumption of diverse single-user photo collection applications that one user takes photographs from only one event per day is acceptable. However, the recall is rather low; we conclude from this that many events are spread over a longer period than 24 hours. It is obvious that they are not captured by these baselines.

- A suitable candidate retrieval step is very important. Comparing our approach or the one from Becker to an approach with a candidate retrieval step, we can see that candidate retrieval has a huge impact on the performance in terms of quality. The approach without candidate retrieval is clearly outperformed by the other approaches incorporating such a step.

- It is surprising that the candidate retrieval step not only increases the efficiency of the system, but also increases the classification performance per se. This is due to an elimination of noise that might confuse the classifiers in the scoring and ranking as well as the new event detection steps. It also shows that the candidate retrieval indeed is able to eliminate many events that the document is unlikely to belong to.

- In comparison to the approach of Becker with a candidate retrieval component, our method has a much higher precision (0.951 vs. 0.909), which is an advantage in our general clustering setting. Our assumption is that a user is interested in seeing pure clusters with only few spurious examples. Having a high precision allows for using a post-processing strategy that aims to increase the recall by merging clusters of similar events. As the increase in recall always comes at the expense of reducing precision, a higher precision as obtained by our approach in comparison to the one of Becker is beneficial.

Moreover, we observe that the performance of *Becker* is much lower than Becker, Naaman, and Gravano [BNG10] report in their publication. This difference can be explained by the fact that the number of documents to be clustered differs. While we tested our system with 131,211 samples, Becker et al. used a dataset consisting of only 27,000 documents. When using more documents, there are also more events. As a consequence, the chance of assigning a document to a wrong event is much higher. This explains the lower F-measures.

6.6 Conclusions

In this chapter we have presented our single-pass approach for clustering with the event-based stream classification framework. We indeed have shown that our system is able to classify a stream of social media data into a growing and evolving set of events. Our method has been applied to our ReSEED dataset allowing for an easy comparison with other approaches.

In particular, we have shown that our framework with the single-pass approach is able to successfully address the following key problems:

- We are able to tackle the new event detection problem, i. e. the problem of determining whether an incoming data point belongs to a new event or not.

- We are able to scale to the data sizes and data rates which are encountered in social media applications.

These problems were addressed successfully by including a candidate retrieval step retrieving a set of event candidates that the incoming data point is likely to belong to and by including a function that was trained using machine learning techniques to determine whether the incoming data point belongs to the top scoring candidate or rather to a new event.

We have directly compared our system to several baselines showing that it outperforms all of them in terms of F-measure. Our approach also outperforms the approach from Becker et al. in terms of efficiency. The benefit of our approach is that it has the capability to scale. The processing time per document remains nearly constant with increasing documents, addressing the scalability challenge mentioned before.

The result of our system with 88.6 % F-measure is already good, but nevertheless we acknowledge that the UserDay baseline also reached a high value at very low computational cost. We see that the baseline performs better in terms of precision leading us to investigate further into the direction of increasing recall. We intend to examine whether the performance of our approach can be increased by the use of a second pass which merges different events into one. This will be investigated in the next part of the dissertation.

PART III

Multi-pass Stream Clustering

CHAPTER 7

System Description of the Stream Classification Framework for a Multi-Pass Setting

In the previous part of this dissertation, we presented and experimentally evaluated our stream classification framework used with a single-pass strategy. We have shown that our framework with the three components candidate retrieval, scoring and ranking, as well as new event detection is indeed able to cluster a data stream of social media documents successfully. The advantages of the system over other state-of-the-art approaches are that a) it is scalable to the data sizes and rates needed in the area of social media applications, allowing even to scale to data streams of infinite size—theoretically—, assuming that all events and documents can be stored physically, and b) it has a very good overall performance which outperforms several baselines and competing approaches on our real-world and non-toy ReSEED dataset.

Even though the performance and results of our single-pass approach are already very good, we are aware of other approaches using our dataset in a non-stream context showing a better performance (for more information see also Section 3.7). This proves that it is still possible to reach better results. We discovered that not only our approach but also two simple baselines are able to produce very high values for precision while keeping a high overall performance in terms of F-measure. This is interesting for us in two ways.

First, we assume that users are interested in seeing very pure clusters with only few or in best case no noisy elements. To clarify this, let us assume two different scenarios. In the first scenario, a real event is split into three distinct events by an algorithm; in the second scenario, all documents of this event are merged correctly into one single cluster but in addition other documents are merged into that cluster that is not belonging there. In the first case, a user can easily identify that the distinct events actually depict only one event and they are finally merged resulting in a still pure and desired event cluster. Cleaning up the cluster produced in the second case involves a lot more work as every element has to be checked if it belongs to the cluster or not. We clearly see that the first case can be tackled easier.

Second, more importantly, if it is possible for humans to merge the resulting event clusters to reach a better overall clustering. This merging process is also possible for a machine learning system. Therefore, our idea is to apply another classification process on top of the resulting clustering. As our single-pass approach and the two baselines actually produce results similar to the first case of our example above, we indeed have a very good starting position to merge the produced clusters further. Practically, the strategy is to increase recall by merging clusters depicting similar events. It might be possible that an increase of recall comes at the expense of precision. Nevertheless, if the precision is lowered only slightly in turn of a large increase of recall (with the result of a very good overall performance), this is an actual improvement from the perspective of the end-user.

Therefore, in the following, we investigate how our approach can be improved by additional classification passes so that the recall of the final clustering is increased. We thus aim at refining our stream classification framework further with the goal of a better overall performance. Therefore, in the following two sections of this part, we are concerned with the following problem statement:

Problem 7.1 Given a highly precise pre-clustered data stream of social media document with each source document assigned to one event cluster, we aim at finding a further clustering identifying the final event clusters of the source documents. ■

We are still faced with the challenge of classifying a never-ending stream of data into corresponding events. Nevertheless, due to a preceding clustering step, the number of elements to be clustered is a lot smaller. This has the advantage that less comparisons have to be made, thus saving computational power. The pre-clustering of the data also has the benefit that features which were missing for single documents might be available with the intermediate cluster to which the document is associated. This might facilitate the assignment to the final cluster as more features for the decision can be used. However, the number of the target event clusters is still unknown and the determination of that number is still challenging. But, as we will show, our framework is capable to address these challenges successfully.

In this chapter, we introduce a strategy to extend our *Event-based Stream Classification Framework* to be used for an event classification using a multi-pass strategy. We will show that the extended system is capable to handle the same document set which has been used for the single-pass approach; the results of this approach are therefore fully comparable to those of the single-pass strategy. In addition to that, we will show that the scalability and performance of the classification system is better when using two passes. In general, a theoretically unlimited number of passes is possible and we will present an overview on how this can be realized. Later, we demonstrate this approach using the example of a two-pass strategy.

7.1 Problem Statement

As already stated in Section 5.1, we view the problem of event detection in social media as a supervised classification problem where the underlying similarity function and other clustering algorithm parameters are optimized using labeled data. Doing a reclustering of an intermediate clustering is actually not different to the task of clustering the original documents. The produced set of intermediate clusters can be handled as if it were sets of documents. Consequently, the problem we are facing in this chapter actually stays the same.

Instead of a set of documents D, the input of our framework is a pre-clustered set of D. We call such a pre-clustered set an intermediate clustering and denote it by C_ι. In analogy to our problem in the previous part, we still need a function a_t which assigns the intermediate event $c \in C_\iota$ to one of the newly created event clusters E_ν at time t. This function is defined as following:

$$a_t : C_\iota \rightarrow E_{\nu_t} \tag{7.1}$$

The new events E_ν are considered to be the resulting clusters. These clusters can then again be used as starting point so that $C_\iota = E_\nu$. Actually, the function a_t can be executed recursively until the desired granularity of clusters is reached. In each step the number of newly created events is unknown, and we thus depend on a similarity function sim. We keep the similarity function to be learned similar to the one used in the single-pass scenario (see Equation (5.3)):

$$sim : C_\iota \times E_\nu \rightarrow [0..1] \tag{7.2}$$

Depending on the value sim and a therefrom deduced feature vector $\vec{v}_{new}(c)$, we decide whether a new event is created or if the function sim assigns the intermediate event clusters to the event cluster e_ν by maximizing the similarity:

Definition 7.1.1 — Assignment function

$$a_t(e_\nu) = \underset{e_\nu \in E_\nu(t)}{\arg\max}\, sim(c, e_\nu) \tag{7.3}$$

The similarity between an event cluster c of the intermediate set C_ι and a cluster e_ν of the new event set $E_n u$ is calculated using the following linear model:

Definition 7.1.2 — Similarity function

$$sim(c, e_\nu) = \vec{w} \cdot \vec{v}_{sim}(c, e_\nu) \tag{7.4}$$

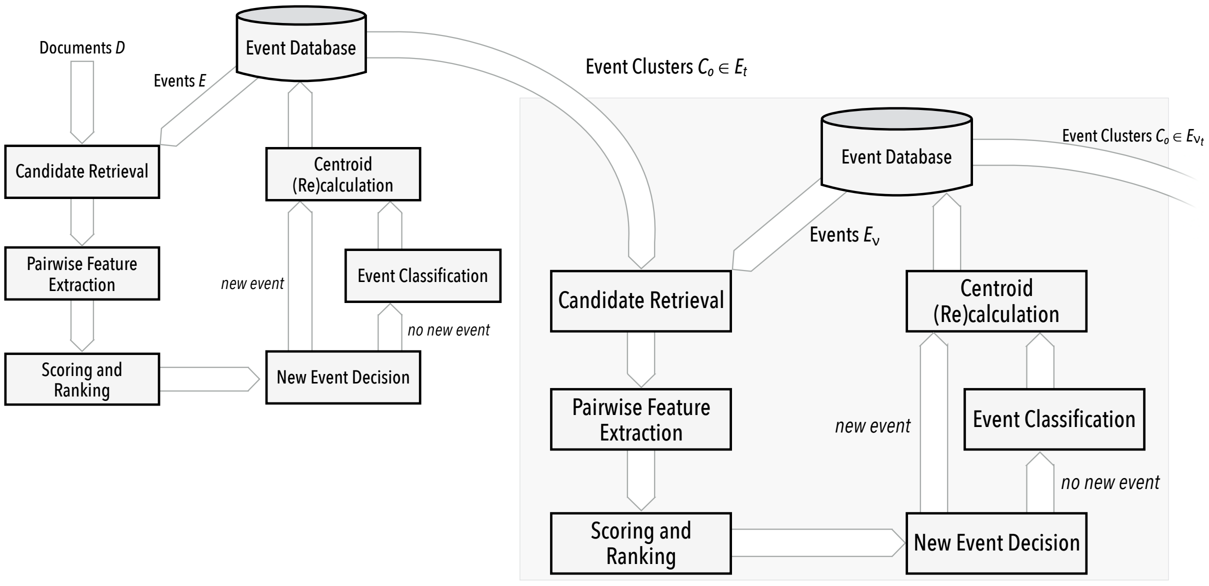


Figure 7.1.: Overview of the event clustering framework system using multiple passes

Again, the already known features like time, location, tags, etc. are used with the same simple similarity measures for the different dimensions of the vector $\vec{v}_{sim}$. As the task remains to optimize the weight vector $\vec{w}$, we see that the approach itself does not change and the only difference is the initial item set C_ι to be used.

The main problem is shifted towards the question at what position in the classification process we can start with the next pass. In a stream setting it is not possible to wait until the first classification pass has been finished, as this will never be the case in a stream data setting. Fortunately, it is not necessarily needed that the whole set has been clustered before the next classification pass can be started. As the data is ordered by time, the intermediate event set could be handed over to the next pass if the timestamp of the current data point in the classification process has passed a certain point in time. An alternative to this is to allow the shift of the intermediate event set to the next pass after a fixed number of data items processed.

7.2 System Overview

In this section we describe how and in which ways our framework can be used in a multi-pass setting. The first possibility is to use the framework in a cascaded mode as shown in Figure 7.1.

Our system processes an incoming data stream of documents D in several steps starting with the process already described for the single-pass mode, with the difference that there is a time-based constraint. In detail, such a cascaded system works as follows:

1. Each document $d \in D$ is processed using the candidate retrieval, scoring and ranking, new event decision, as well as event classification steps. This process runs infinitely processing the document in timely-ordered manner as provided by the data stream.

2. If a certain time-based criteria is fulfilled, a snapshot of the event database of the first processing step is taken. All the events at a specific time t are added to the input cache C_ι of the second process chain. The copied events $C_\iota \in E_t$ are then deleted in the first event dataset so that $E = \varnothing$.

3. The second clustering process starts when the first pre-clustered event sets are added to the item queue C_ι of the second process. Each cluster $c \in C_\iota$ is then processed in timely order using again the single steps of the process (candidate retrieval, scoring and ranking, etc.).

4. As many clustering passes as desired can be added repeating steps 2 and 3.

An alternative setting is to use a different clustering strategy to produce the intermediate event clusters. Any clustering strategy can be used to achieve this intermediate clustering as long as it can be used in a stream data setting. For example, it would be possible to delay the data stream by one day and cluster the documents within that day by user. Then this intermediate clustering could be given to the classification system.

The adding of additional passes to the clustering process with our framework makes sense as long as the precision of the resulting clustering is sufficiently high. This allows for an increase in recall.

7.3 Multi-pass Requirements and Challenges

Using a multi-pass strategy, there are some requirements and challenges which we need to pay attention to: i) achievement of a better overall performance, ii) keeping the system scalable, and iii) sticking to a stream paradigm. Our goal is to find a suitable strategy which fulfills all these criteria.

The challenge to achieve a better overall clustering performance in comparison to the single-pass strategy depends on different factors. In general the features used for the classification process must allow for a better clustering. Given that this is the case, a better performance can be reached if the intermediate clustering of a previous pass is pure enough allowing for a further merging of the intermediate event clusters without a too great error rate. Thus, we mainly depend on the intermediate clustering created in a previous classification pass. If this clustering does not provide a reasonable starting point, a better clustering is hard to achieve. In general, a previous clustering enriches the newly created single data points with more condensed information. Therefore, in our case, it shall be possible to reach a significantly better overall result as the intermediate clusters produced by our single-pass strategy and several baseline are of high precision.

Scalability is still crucial for a system tackling the problem of clustering real-world social media data. The use of multiple passes is insofar problematic as every pass needs computational power. Consequently, the scalability is lower with each pass which is added to the clustering process. We can cope with this issue by ensuring that the single passes can be done efficiently. Therefore, one strategy to tackle the scalability challenge is to use classification techniques allowing to produce a coarse clustering very quickly as a first pass. After that, another pass is used to classify the intermediate event cluster set (which obviously is smaller) more thoroughly.

Sticking to the stream-based approach is a crucial aspect because of the stream-based nature of the data. The usage of approaches for the clustering process is thus limited to techniques which allow to process data in a stream-based manner. However, we are aware that a stream-based classification in real-time is only possible using one single pass. Nevertheless, it is possible to use more than one pass if data items are cached for a very short period of time. Which caching time is acceptable depends on the intended purpose.

Ideally, a multi-pass strategy, i) uses as few passes as possible, ii) finds a way to increase recall while keeping a high precision, iii) is faster than a single-pass strategy, and iv) finds a caching strategy which is acceptable for the task.

7.3.1 Number of Passes

The number of passes does not only influence scalability, but also the way a multi-pass system can be trained. Actually, the classifiers used in each pass have to be trained with a training set adapted to that pass. The criteria based on which the data can be classified changes in each pass. We thus need to train different models so that they suit the actual classification situation.

Getting good and suitable training data gets more difficult with each pass. The reason for this lies in the (unwanted) noise which occurs with every clustering pass. It has to be taken into account that the noise produced in a previous step has also influence on the training of the classification model in the current pass. It is not possible to use only clean training data for every pass as this does not reflect the actual situation in the pass and eventually leads to a bad quality of the overall classification process. Therefore, it is our aim to limit the total number of passes to two.

7.3.2 Influence on Framework Settings

As we have discussed above, the training data for the classifiers in the different passes varies. Also, the settings of different steps of our clustering framework have to be readjusted so that they are optimized for each pass. So, we have to look again at all three steps of our framework: candidate retrieval, scoring and ranking, and new event detection. It has to be determined whether the settings chosen for the all of these in the single-pass strategy can be adopted or if these setting have to be adjusted.

Candidate Retrieval

As with the single strategy, we keep the candidate retrieval step to ensure a scalability of the overall system. We take a look at the most promising candidate retrieval strategies and compare them regarding their effectiveness as defined in equation (5.5). In particular, the strategies compared are the following ones:

1. **k-nearest by capture time**: By using this strategy, we retrieve those k events with the lowest temporal distance to the document. For this, we order all events e_i in the event database by $\Delta(\text{time}(d), \text{time}(e_i))$ and then return the top k events in this ordered list.
2. **k-nearest by upload time**: We follow the same strategy as in *k-nearest by capture time* but use the upload timestamps.
3. **Tag-TFIDF**: We score each event by summing up the TF-IDF values of every tag which the document and the event share and return k events having the highest scores.

Scoring and Ranking/New Event Detection

For the scoring and ranking step, we learn the likelihood $P(e_\nu|c)$ that an intermediate event cluster c belongs to a given final event cluster e_ν. This likelihood is computed using a classification algorithm. As the different types of Support Vector Machines have been shown to work quite similar, we only compare a standard SVM to Decision Trees. The probability is calculated using the distance to the decision boundary when using a SVM. The decision tree probability estimates are a calculation from the class frequencies at the leaves. The problem formulations using both classifiers are the same as given in Section 5.5.

For the new event detection, we use a classifier for the decision whether an intermediate event cluster belongs to an existing final event cluster or if a new final event has to be created. Here, we also employ a standard Support Vector Machine. As we have discovered the best features for such a decision in Section 6.4, we adopt these features for the decision model in the multi-pass setting. Therefore, the feature vector $\vec{v}_{new}(c)$ used for the new event detection decision for each intermediate event cluster c looks as follows:

$$\vec{v}_{new}(c) = \begin{pmatrix} \text{max} \\ \text{avg} \\ \text{stddev} \\ \text{max}_{\text{upl}} \end{pmatrix} \tag{7.5}$$

7.4 Multi-pass Strategies

In this section we discuss different strategies for extending our clustering framework to a multi-pass setting. As we have described before, we limit the number of passes to two in order to be able to produce training data for each pass which is still of good enough quality and thus acceptable for the training of a classifier. Thus, our process will consist of two passes. Consequently, we have to examine which strategy works well for the two passes.

We take a look at several strategies altering the first pass, i. e. the pre-clustering step. In principle, as we have shown in the previous section, the framework is capable to use any pre-clustered item set as long as the data structure matches the one of the original document set. In particular we examine the following strategies for the first pass in terms of quality and performance:

1. **NormalPreclustering**: This strategy uses a single-pass pre-clustering as we have presented in Part II. The intermediate event cluster is moved to the second clustering if it is not returned by the candidate retrieval step at the arrival of a new document. The delay caused by this strategy is variable depending on the time-based density of the event clusters. Therefore, the initial clustering quality in terms of F-measure is 0.886.
2. **NormalPreclusteringByDay**: In this strategy we essentially do the same as in the first strategy with the difference that the clusters are moved to the second clustering process day-wise. If a document arrives in the clustering process having a different upload day than the previous documents, the event set is emptied for the first clustering pass and the event cluster of the old day are passed to the second pass. This limits the event length in the first pass clustering to 24 hours.
3. **SimpleUserDayPreclustering**: The dataset is clustered by user and day. This is a very cheap operation which can be done very efficiently using an indexed structure. As in the previous strategy, the clusters produced are then given to the second pass day-wise. The event length is therefore also limited to 24 hours.

In every two-pass setting, the second pass always uses our original classification strategy as used in the single-pass setting. But, the parameters of the single steps are adjusted. This second pass is then fed with the clustering outcome of the before-mentioned strategies.

We are not only interested in the cluster quality but also want to know how scalable the approach is. As a measure we use the time needed by the single-pass strategy as a baseline. We set this time as a reference and report the difference for each of our two-pass strategies.

CHAPTER 8

Experimental Setup and Results of Supervised Multi-Pass Clustering

In the previous chapter we have proposed different strategies on how our supervised classification system can be used in a multi-pass setting. In this chapter we present the experimental setup of the supervised multi-pass classification approaches and the optimization of the single steps of the clustering framework. We conclude the chapter with the presentation of the multi-pass clustering results.

Even though our classification framework can be used with a theoretically unlimited number of passes, we have already decided on the number of passes after careful consideration to be two. The main reason for this is the challenge to generate a suitable training set for each pass. As the precision in each pass is unequal to 100 % and the precision will decrease with each additional pass, it is unlikely that the training data generated for a third pass or more are still good enough to increase the overall clustering quality significantly. As a consequence, all our experiments presented in this section are conducted using two passes.

In general, our focus in this chapter is on the second pass. The results of the first pass have been discussed in detail in Chapter 6. Nevertheless, we start this chapter with a discussion of the different first-pass strategies proposed in Section 7.4, giving an overview of the initial situation when starting with the second pass.

8.1 Analysis of First-Pass Strategies

Using two passes involves the uncertainty of which strategy to follow in order to gain the maximum in terms of clustering quality and performance. The problem is to find a first-pass strategy which is optimized in two ways: i) good basis for further merging, and ii) efficiency. A good starting point for further merging is given if the intermediate clustering contains very pure clusters, thus having the noise reduced to a minimum within all clusters. In addition to that, there has to be at least one feature left which is still discriminative enough to allow a decision during a further merging.

Table 8.1.: Results for intermediate clustering using different strategies for the first pass

Method	F_1-Measure	Precision	Recall	Time Consumption
NormalPreclustering	0.886	0.951	0.830	1.0000 (Baseline)
NormalPreclusteringByDay	0.811	0.991	0.686	0.6204 (−37,96 %)
SimpleUserDayPreclustering	0.842	0.996	0.729	0.0007 (−99.93 %)

Therefore, it is important for us to examine what will be the initial situation after the first clustering pass, before taking a look at the two-pass clustering as a whole. This includes how well the first pass performs in terms of quality and scalability. We take a look at all strategies described before in Section 7.4 and analyze how suitable they appear for a multi-pass clustering.

First, we use the different first-pass strategies and let them cluster the test set of our dataset into an intermediate set of event clusters. The results of these clustering processes are shown in Table 8.1. Besides of the quality in terms of F-measure, precision, and recall, we also report on the time consumption of these first-pass strategies in relation to the total computational time consumed by our single-pass classification strategy for a complete clustering of the test set of our dataset; the latter we set to a value of 1.0. This serves as a baseline for the comparison of the computational efforts needed for the intermediate clustering processes as well as the the final two-pass clustering process. We remark that the *NormalPreclustering* strategy matches exactly our single-pass strategy; therefore, the time consumption value for this strategy is also 1.0.

The measured results confirm that the *NormalPreclustering* is the best strategy to produce an intermediate clustering in terms of F-measure. The two other strategies show less performance in terms of F-measure, but both strategies still deliver a decent value. This allows the conclusion that all strategies can solve the clustering task successfully. In addition to that, the results in terms of precision are very high for all strategies. The values for the *NormalPreclusteringByDay* and *SimpleUserDayPreclustering* are even higher than 99 %, confirming the high purity of the intermediate event clusters produced. It is remarkable that the *NormalPreclusteringByDay* strategy achieves the least performance being beaten regarding all measures. Therefore, we assume that *SimpleUserDayPreclustering* would be the better strategy to be used for pre-clustering in the two-pass approach.

In contrast to the general performance, the differences between the strategies are very high regarding the computational time consumption. Compared to the *NormalPreclustering*, our baseline, both other strategies consume less time. This is a good initial situation which potentially allows for further improvement of the overall scalability when using these two strategies in conjunction with the second pass. It is remarkable that the *SimpleUserDayPreclustering* strategy is several orders of magnitude faster than our standard clustering. Its absolute computational time is only a little greater time than one second[1].

[1]The used system was a 2013 Intel Xeon 20-core system clocking at 2.9 GHz with 384 GiB of main memory.

8.2 Gold Standard Preparation for the Second Pass

For experiments we use the dataset presented in Chapter 4. For the first pass, as described in the previous section, we used the dataset with its original gold standard to create the training data. However, this is not possible for the second pass as we need to create a different gold standard from which we can create the training data for the second pass. The creation process of the newly needed gold standard depends on the clustering which was produced by the first pass. One implication of that is that the quality of the new training data for the second pass is influenced massively by the quality of the intermediate clustering in the first pass. In the following we discuss this issue in more detail. After that, we describe how the new gold standard is created. This gold standard is suitable for the creation of the training data that is needed for the learning of the classifiers used in the second pass.

8.2.1 Quality Issues in the Preparation Process

As already stated above, the creation of the gold standard for the second pass differs from the creation process of the one used in the first pass. The quality of the gold standard for the second pass heavily depends on the precision of the intermediate clustering produced by the first-pass strategy. If the precision of the resulting merge process of the intermediate clustering were 100 % precise, the gold standard produced for the second pass would be of the same high quality as the original gold standard. This implies that the precision of the intermediate clustering influences the quality of the second pass gold standard. The reason for this lies in the amount of noise within the clusters which eventually occurs during the first clustering process.

Let us consider a scenario where the first clustering pass produced intermediate event clusters with very high precision—preferably pure clusters. Using the original gold standard, we know without doubt to which final target event cluster these intermediate event clusters have to be assigned to. If a purity of the intermediate clusters is not given and they include a lot of noise instead (e. g. having documents in the intermediate clusters belonging to different correct final event clusters), we have to answer which correct final event cluster the intermediate cluster should be assigned to. If there is too much noise involved in the intermediate clustering, the decision on the assignment to a correct final cluster is not possible without lowering the precision. In that case, a creation of a good training set suitable for the classifiers is difficult.

Therefore, we aim to create the gold standard using data derived from an intermediate clustering process which produced very pure clusters. Taking a look at the results for the intermediate clustering methods in the previous section, the most promising strategy to be used for the gold standard creation is the *SimpleUserDayPreclustering* strategy. Even though the precision is not 100 %, we consider its 99.6 % precision to be high enough to reduce the noise to a minimum. This is corroborated by an analysis of the intermediate cluster purity as shown in the right part of Figure 8.1.

In Figure 8.1, we see that 97.6 % of the intermediate clusters produced are pure, i. e. these clusters do not contain any noise at all. Only 0.4 % of the clusters contain documents from more than two correct final event clusters, but this number is low enough to not negatively

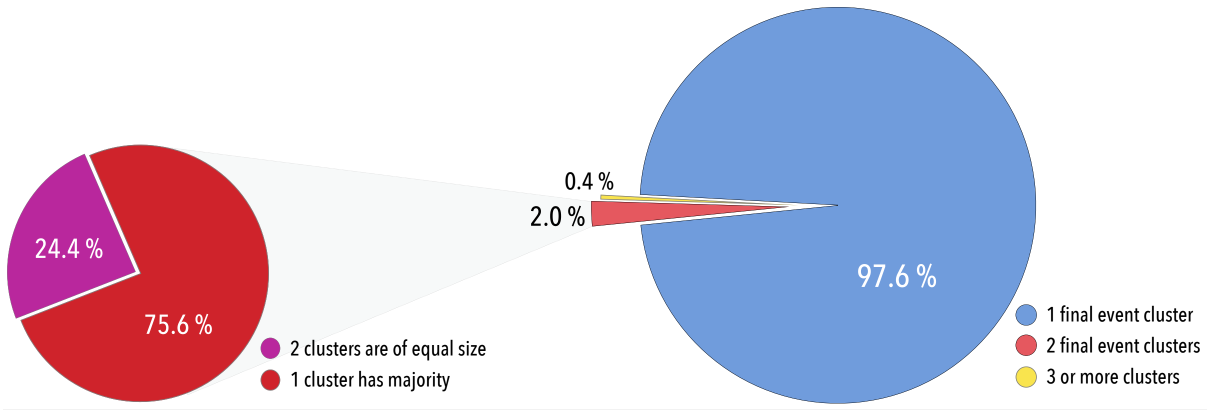


Figure 8.1.: Analysis of final event clusters contained in an intermediate cluster

influence the task. About 2 % of the intermediate clusters consist of exactly two final correct event clusters. We have analyzed this further in the left part of Figure 8.1. About three forth of the "two-event" clusters have one final event which appears more often than all the others. In about 25 % of the cases we are confronted with the situation that the intermediate cluster contain exactly the same amount of documents from two different final cluster; this is mainly the case for intermediate clusters where two single events have been merged mistakenly to one intermediate event. From this analysis we learn that more than 99 % of the intermediate clusters are clearly associable with one final correct event cluster. This licenses the conclusion that this intermediate clustering approach is suitable for the creation of a new gold standard for the second pass.

Another reason to choose the *SimpleUserDayPreclustering* as intermediate clustering strategy is its very reasonable quality in terms of F-measure. Its F-measure does not differ significantly from the ones of the other approaches proposed. We believe that the event clusters produced by all first-pass clustering strategies are similar enough so that there will be no significant differences with respect to the two-pass clustering process when using any of the other strategies in the first pass, even though the classifiers of the second pass are trained using the gold standard based on the *SimpleUserDayPreclustering* and not by another strategy.

8.2.2 Creation of the Gold Standard for the Second Pass

Initially, the training set of our ReSEED dataset is given to the *SimpleUserDayPreclustering* strategy. The outcome of this process is a set of intermediate event clusters. These intermediate event clusters are then used for further processing.

The first step after the production of the intermediate clustering is an assignment process where all produced intermediate event clusters are assigned to one of the original event clusters as defined in the original gold standard. This process is not exact as there is the possibility that an intermediate cluster consists of documents that belong to two or more different original event clusters. In order to handle these cases, we use the following methodologies to create our mapping of the intermediate event clusters to the final event clusters. For each intermediate cluster in the resulting set we do the following:

- We associate a list to the intermediate event cluster in which we store possible candidates of original event clusters to which it could by assigned.
- For each document in the cluster, we look up the original mapping to its original event cluster. If a new original event cluster is found, we create a new entry for this event cluster in the list. If the original event cluster is already on that list, we increment its count by one. Eventually, we end up with a list of all possible original event clusters together with the number of their appearances.
- The list with the possible original event cluster candidates is reordered in descending order by the number of their appearance. If two or more original event cluster candidates have an equal number of appearances, they are ordered in ascending order by their first appearance in the original gold standard.
- We associate the intermediate event cluster to the original event cluster being on top of the candidate list.

The created gold standard for the second pass finally includes 20,835 intermediate event clusters. It can now be used for the creation of the training data for the different steps of the clustering framework for the second pass. This will be described below.

8.3 Optimization of the Classification Framework Steps for the Second Pass

In this section we take a look at how the single steps of the classification framework can be optimized in order to improve the overall classification process. While many aspects are adopted from the settings used in the single-pass scenario, some settings have to be retuned in order to accommodate the slightly different setting. Here we discuss the optimization experiments which we conducted on each part for the clustering framework. Later, we incorporate an exhaustive search to find the best overall settings optimized for the final task.

8.3.1 Candidate Retrieval

In Section 5.3 we described different strategies to retrieve candidates for the single-pass strategy. This step is used to make the approach scalable as the newly arriving documents do not have to be compared to every event in the event set but only to the ones proposed by the candidate retrieval step. In contrast to the single-pass strategy, the number of items to be processed is lower in the two-pass strategy, making the candidate retrieval step less crucial. However, since we have shown that the overall classification performance is higher when a candidate retrieval strategy is used, it is advisable to use such a strategy also in a two-pass approach.

So, as in the single-pass setting, we have to find the minimal number of candidates among which the correct candidate is included. We therefore compare the different candidate retrieval strategies defined in Section 5.3.2. We only choose strategies which have been shown effective in the single-pass process. Before we present this comparison, we give an overview of the experimental settings of this optimization step.

Experimental Settings

Our objective is to find out how effective each candidate retrieval strategy works. This is similar to the analysis presented in Section 6.2.1. However, here we use the new gold standard for the training set of our dataset as a basis again. We make use of our simulation of an optimal clustering. This is an iterative classification process where we use the new gold standard information to assign the intermediate clusters to the final correct event. For more details see Algorithm 6.1 on page 88.

Using the clustering simulation we search for the rank of the correct event cluster for each intermediate event cluster. As a result we get the number of events to be retrieved so that the correct event for the document is included. Using these numbers, we obtain the effectiveness for a certain number of candidates k as defined in Equation (5.5) in Section 5.3.1 for the different candidate retrieval strategies.

Results

In the following, we report the results of the effectiveness for the single-strategy candidate retrieval strategies. The candidate retrieval strategies considered are the retrieval by upload an capture time. We present the effectiveness of these strategies over the number k of retrieved events in Figure 8.2.

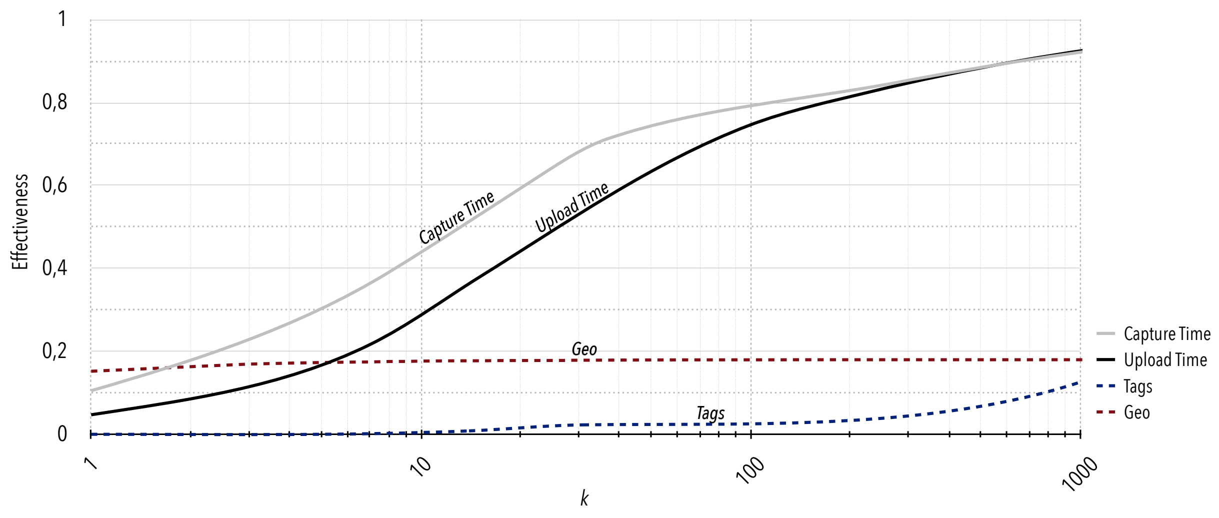


Figure 8.2.: Effectiveness of different candidate retrieval strategies used for a second pass

Table 8.2.: Needed number of k to reach x % effectiveness

Candidate Retrieval Strategy	90 %	70 %	50 %	30 %	20 %
Upload Time	611	74	27	11	7
Capture Time	733	34	13	5	3
Geo	–	–	–	–	–
Tags	–	–	–	3673	2102

The graphs clearly show that only the candidate retrieval strategies based on the temporal features are able to reach an effectiveness level exceeding 90 %. In fact, both candidate retrieval strategies based on a temporal feature can reach an effectiveness close to 100 %. However, this is only the case if a very high number of documents (over 1,000) is retrieved. This is undesireable as the overall computational cost for further processing steps is too high for such an amount of candidates.

On the other hand, the other two candidate retrieval strategies only reach a very low effectiveness making their usefulness questionable. While at least the candidate retrieval strategy based on the geographic location reaches its maximal effectiveness very quickly with only few candidates retrieved, the candidate retrieval strategy using tags has serious problems to reach any effectiveness at all for a low number of candidates. Therefore, we do not employ any strategies combining those single feature retrieval strategies.

In Table 8.2 we outline the number of event candidates which have to be retrieved by the candidate retrieval strategies in order to reach an effectiveness of 20 %, 30 %, 50 %, 70 %, and 90 %, respectively. In comparison to the candidate retrieval strategy effectiveness in the single-pass strategy (see Table 6.1), more candidates have to be retrieved in order to reach the same effectiveness level. For example, to reach an effectiveness of about 80 %, we needed to retrieve only 6 or 7 candidates using the capture or upload time strategy, respectively. With the two-pass strategy, reaching the same level implies to retrieve already more than 100 candidates.

We learned from the single-pass strategy that the impact of a candidate retrieval strategy on the final performance of the system is different than it appears in the effectiveness analysis. Therefore, we take a look at the retrieval strategies as well as the number of candidates to be chosen once again later. As the retrieval strategies *Geo* and *Tags* have strong limitations regarding their maximal effectiveness even for higher candidates k, we only consider the retrieval strategies *Upload Time* and *Capture Time* for further experiments.

8.3.2 Features for Similarity Function Learning and New Event Detection

Having finished the candidate retrieval step and extracted the pairwise features for all relevant combinations of an intermediate event cluster and a final event to which it corresponds to, we end up with feature vectors of single similarities. Next we train a classifier which is capable to use these feature vectors to provide us an overall similarity measure of candidate pairs. The learning strategy and data used for this is discussed next. After that we also take a look at the new event detection used in the two-pass approach.

Similarity Function Learning

In the following we discuss what techniques we use to learn a similarity function in the two-pass strategy. In order to learn a similarity function we need suitable data. Therefore, we first discuss the creation of suitable training data. After that, we describe how we conduct our experiments to decide on the settings and options used for the used classifiers for the scoring and ranking as well as the new event detection steps.

Creation of the Training Data

In order to create the training data we make use of the new gold standard for the second pass described earlier in Section 8.2.2. The basis for the training data creation was the intermediate clustering produced by the *SimpleUserDayPreclustering* strategy when applied on the training set of our original dataset. We base our strategy on the findings during the training process of the similarity functions used in the single-pass setting. In particular, we create the training data using the following steps:

- For each intermediate event cluster c and each centroid representing one final event e_ν, we compute the corresponding similarity vector $\vec{v}_{sim}(c, e)$. The pairs $(c_i, e(c_i))$ consisting of an intermediate event cluster c and the corresponding final event $e(c_i)$ are used as positive examples.
- For the positive examples, we choose n consecutive intermediate event clusters from the training set (which is ordered by time). More specifically, we use the first documents appearing in temporal order.
- Regarding the negative examples, we use the nearest sampling strategy as proposed in Section 6.3. These examples are chosen using the final event cluster with the highest similarity to which the intermediate event cluster does not belong:

  $$neg(c_i^+) = \max_{c_i^- \notin ext(e(c_i^+))} \sum_{i=1}^{6} (sim_i(c_i^-, e(d_i^+))$$

  .
- We take care that the number of positive and negative examples is equal so that the training data set is balanced.

As we have already found the optimal number of training examples to be used for the training of the classifiers (see Table 6.2 and Table 6.4 in Section 6.3.2), we make use of these values. Therefore, for the training of the Support Vector Machine we can use all 4,550 positive together with 4,550 negative examples. This is the maximum number of positive examples which can be extracted including all single features used for the classification. For the training of the Decision Tree we use as many training examples as for the SVM. Here, due to the limited number of training examples, we clearly see that a classifier needing less training examples is indeed an advantage. If not all features are used, the number of training examples is higher, e. g. the number of training examples when not using the geographical feature is 11,938.

Classifier Settings

The settings for the different classifiers are comparable to the settings used for the single-pass strategy. We employ the following settings for the two classifiers SVM and Decision Tree:

- **Support Vector Machine**: We employ the C-SVC implementation for support vector classification from LibSVM[CL11]. This is a standard SVM. We set the trade-off between training error and margin C to 1.0 and make the classifier output so-called probability estimates. This provides a normalized probability value in the interval between 0 and 1 calculated from the distance of the data point in the vector space to the decision boundary. We use this probability as our actual similarity measure. We experiment with a linear and a RBF kernel.
- **Decision Tree**: We use the Classification and Regression Trees (CART) implementation following Breiman et al. [Bre+84]. The maximum depth of the Decision Tree is set to 5 to avoid a possible overfitting. The minimal samples to be included in one leaf is set to 1.

New Event Detection

The goal here is to train a classifier model for the detection of new events. As we have described earlier, we employ a feature vector $\vec{v}_{new}(c)$ including features that are derived from the ranking and scoring step. In order to use these feature vectors for classification, we also need to choose classification algorithms to induce a model. Therefore, the creation of the training vectors are described next. After that, we describe the classifiers that are used. In addition to that, we describe the settings and configuration used for the employed classifiers.

Creation of Training Data

In order to create the training data for the new event detection step in the second pass, we use the intermediate clustering of the training split from our dataset. The total number of intermediate clusters is 20,835. These intermediate clusters are spread over 14,882 final clusters. As a consequence, the maximum number of training data vectors for the new event detection is limited to 14,881 positive examples.

We use an oracle, a clustering simulation which produces an optimal clustering according to the correct assignment of intermediate event clusters to final event clusters. In this process, the similarities of the intermediate and final event clusters are calculated using the learned similarity measure from the scoring and ranking step. For this we use different classifiers and strategies regarding the kernel (in case that an SVM is used). The intermediate clusters are assigned to the correct final event cluster independently of that decision.

The candidate retrieval strategy also needs to be involved as this is important for the calculation of the average and standard deviation features of the new event feature vector. Therefore, we use the most promising candidate retrieval strategy as found in the effectiveness experiments: the *Capture Time* strategy. We configure this strategy using $k = 10$. Thus, the 10 nearest events based on their capture timestamps are retrieved for each intermediate event cluster.

After the process has finished, we end up with 12,690 positive training examples. We randomly choose another 12,690 negative examples from the negative examples which were produced in the optimal clustering process. These vectors can now be used to train our new event detection classifiers.

Strategy for Classifier Training and Classifier Settings

In the single-pass strategy we only used a standard Support vector Machine as classifier for the new event detection step. For our experiments in the multi-pass paradigm we not only use a standard SVM but we also employ a Decision Tree to take a look at the impact of this classifier on the new event detection step.

The actual implementations used for both classifiers are the ones used before (LibSVM and CART). We employ a C-SVM that is used together with an RBF kernel. The parameter C for the trade-off between training error and margin is set to 1.0. The configuration of the Decision Tree is set so that the maximum depth is limited to 5 and the minimum number of elements per leaf is 1. The output of both classifiers is configured so that a linear value in the range [0..1] is returned. The actual value denotes the probability with which the data point belongs to the positive class.

The SVM is trained with 2,000 positive and 2,000 negative examples; these were the optimal number of training examples in the training phase of the single-pass strategy. In contrast to that, the Decision Tree is trained with the maximum number of training examples available (12,690) as we are aware that many training examples are needed in order to obtain a good decision model with this classifier.

8.4 Clustering Framework in Two-Pass Mode – Optimization

Previously we have discussed the methods and settings to be used to optimize each single step of our clustering framework. Before we provide the final results, we need to discuss how the single steps can be optimized globally so that the interaction between these steps results in an optimal classification process.

We learned from the optimization of our system with one single pass that the usage of the best settings for each single task of the overall system does not necessarily result in a system reaching the best result on the overall clustering task. As a consequence, we here outline our findings for the settings of the single system parts which lead to a system that suits best on the overall task in two-pass mode. We start with the description of our exhaustive search allowing to find the optimal parameters for the scoring and ranking as well as the new event detection step. After that, we optimize the candidate retrieval step. As already mentioned before, we always use the *SimpleUserDayPreclustering* strategy for the first pass as a basis.

8.4.1 Exhaustive Search for Optimal Features in Scoring, Ranking, and New Event Detection

It is our goal to optimize all classifiers and their settings so that the overall performance is optimized. Therefore, we do an exhaustive search using the training split to find the best working configuration. Using the single-pass strategy we only compared system configurations that were likely to show a good overall performance as the computational effort was too high for an exhaustive feature search. In contrast to that, using the multi-pass strategy we are capable of searching over many feature dimensions. This includes the scoring and ranking as well as the new event detection step. Actually, we do the exhaustive search using all of the following dimensions at the same time:

- **Similarity Learning Classifier**: We look for the best performing classifier. The classifiers in question are a standard Support Vector Machine and a Decision Tree.
- **Similarity Learning Features**: By examination of this dimension we aim to identify the best combination of single similarity features to be used for the training and prediction of the overall similarity between an intermediate event cluster and a final event cluster. The actual single similarity features included in the search are: sim_{upload}, sim_{capture}, sim_{geo}, sim_{tags}, as well as sim_{title}.
- **New Event Detection Classifier**: Using this dimension we aim to determine the best performing classifier used for the prediction of new events. The classifiers being tested are a standard SVM as well as a Decision Tree.
- **New Event Detection Features**: We analyze the features used for the new event detection decision and search for the best combination to be used for the new event decision. The features being examined are the following: max, min, avg, $stddev$, max_{upl}, and max_{cap}.
- **New Event Detection Threshold**: This dimension is used to find the optimal threshold for the decision when a new event has to be created or not.

Using this exhaustive search we compare 2^{13} different system configurations for the first four dimensions. Every single configuration of the 2^{13} configurations is analyzed individually to find the best threshold for the new event detection decision. In order to find the best-performing value, we employ a gradient descent strategy. As a result we get the optimal threshold for each configuration. Overall, we computed values for over 80,000 individual configurations.

As the exhaustive search strategy uses the complete classification process to compute the final values in terms of precision, recall, and F-measure, we have to provide a configuration for the candidate retrieval step as well. Unfortunately, the candidate retrieval step could not be integrated in the exhaustive search due to computational time restrictions. Therefore, in the optimization step we specify the candidate retrieval strategy to be the *Capture Time* strategy

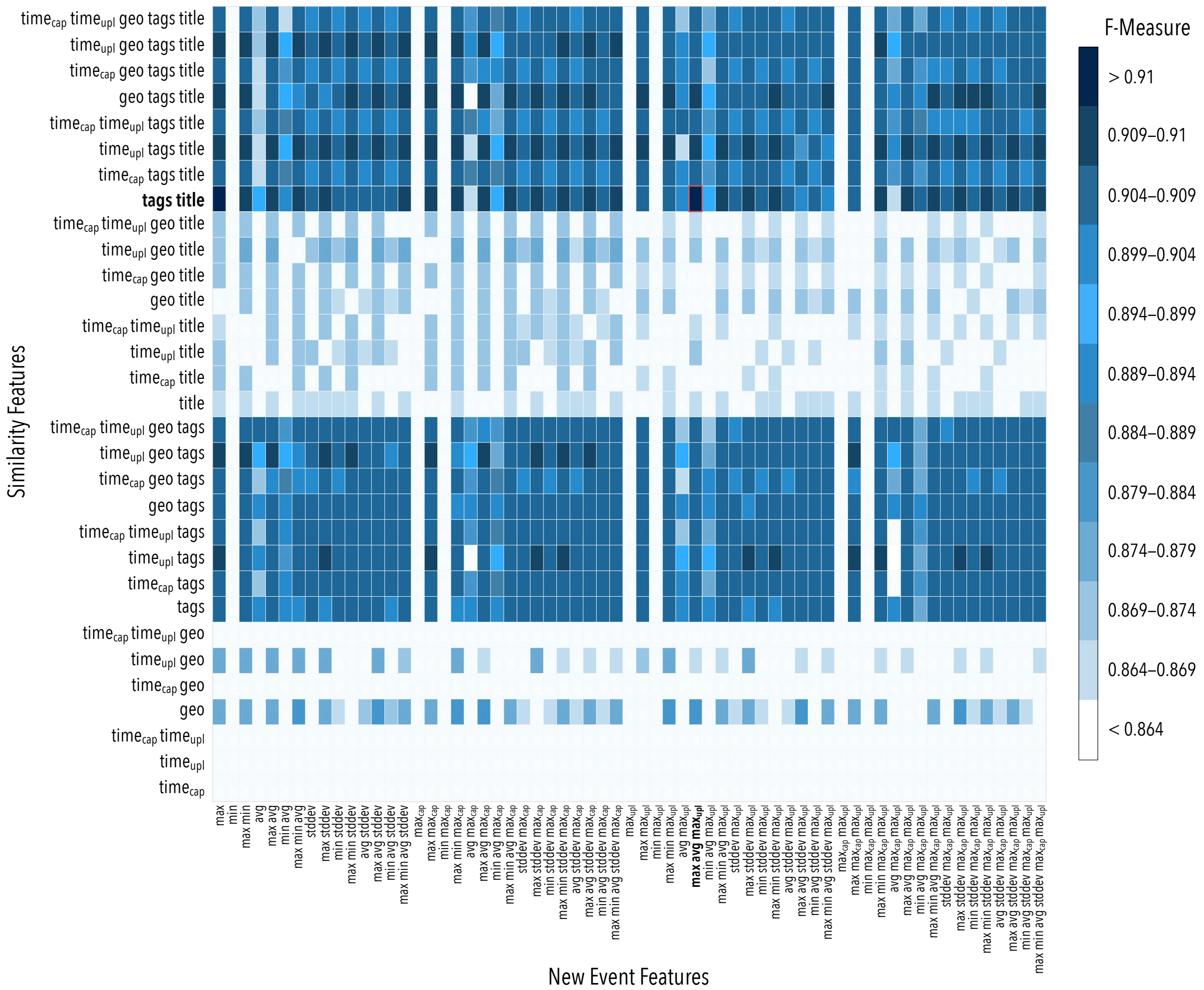


Figure 8.3.: Heat map showing overall performance of the system using different similarity and new event features using best threshold for new event detection (SVM)

and set $k = 10$; thus, 10 final event candidates are retrieved for each intermediate event cluster. We use this number of candidates to allow for an exhaustive search.

All experiments are conducted on the training set of our dataset. The exhaustive search is done using the complete training set and each configuration clusters the whole set.

8.4.2 Results of the Exhaustive Search

In this section, we present our findings of the exhaustive search in the five dimensions mentioned before. Therefore, we have plotted the results in two heat maps. They are shown in Figure 8.3 and 8.4.

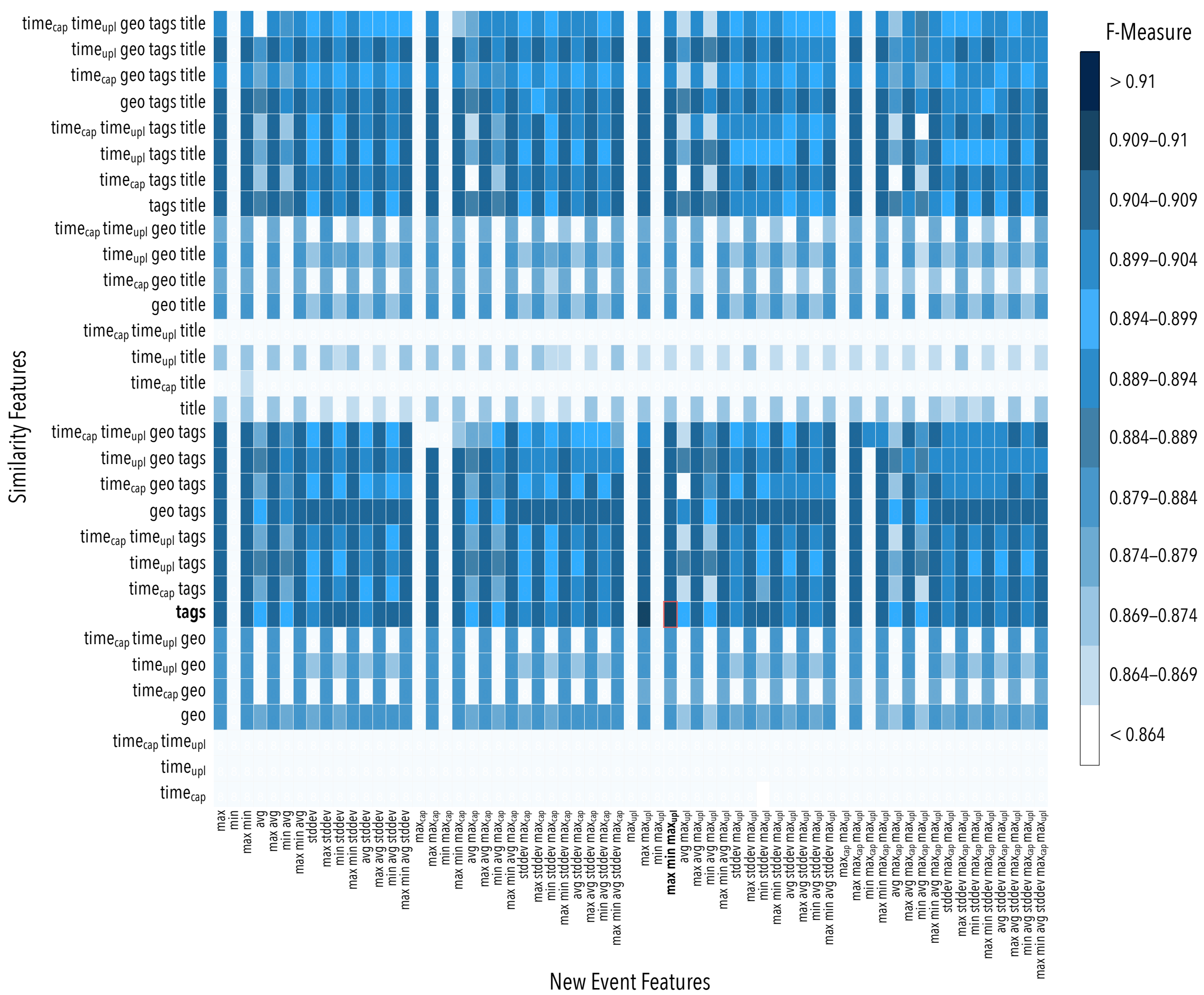


Figure 8.4.: Heat map showing overall performance of the system using different similarity and new event features using best threshold for new event detection (Decision Tree)

In both heat maps, the individual points in the matrix represent the results for one final clustering in terms of F-measure. On the x-axis, we plot the feature combinations which were used for the new event detection classifier. On the y-axis, the single similarity features used for the training and prediction of the overall similarity classifier are depicted. The individual values for the overall performance shows the best results achieved using the best new event detection threshold that has been determined for the feature combinations in question. The graphic in Figure 8.3 plots the results when using a standard SVM as both classifiers. The graphic in Figure 8.4 shows the results when using a Decision Tree as classifier for both classification problems.

Overall, we see that both classifiers can be used to solve the problem. The best performing settings for the Support Vector Machine result in an overall performance of 91.0 % in terms of F-

Table 8.3.: Best performance on overall task on training set using different similarity features and similarity feature combinations

Similarity Features	SVM	Decision Tree
Time_{upl}	0.861	0.861
Time_{cap}	0.862	0.863
Geo	0.882	0.881
Tags	0.908	**0.909**
Title	0.865	0.871
Tags + Title	**0.91**	0.907
All	0.907	0.902

measure. Using Decision Trees as classifiers results in a slightly lower performance (F-measure: 90.9 %), but the differences are minimal. We did not plot the cases where one classification decision is made using a Support Vector Machine and the second using a Decision Tree or vice versa. The results of these combinations reach comparable results to the one using only one classifier type for both decision problems (both combinations reach a F-measure of about 90.9 %).

It is more interesting to take a look at the features to be used for the decisions, since they are different depending on the used classifier. Regarding only the single similarity features used as overall similarity measure, we observe that some of the single similarity features are indeed better than others in helping to solve the problem. The values in terms of F-measure for the single features in question as well as the best performing combination for both types of classifiers are shown in Table 8.3. There we can observe that the usage of the time features similarities is not very effective. The reason for this lies in how the intermediate clustering has been created; here, the time information was used and it is thus less effective for further merging. As both time features are no more effective for solving the overall problem, we conclude that there is a direct statistical dependency between the two timestamps. Other features are more effective, e. g. using the title information allows a slightly better performance. Nevertheless, this feature only has very low impact if used alone and not in conjunction with other features.

The other two remaining features incorporating geographic information and tags look more promising. Nevertheless, we also see great differences in the performance of both. We see that the second-best similarity feature is geographic location. This feature has its limitations as many intermediate event clusters (about 55 %) still lack this feature even if an intermediate clustering has been created. This explains very well why the performance increase is limited when using this feature.

Far and away the best single similarity feature is the one based on the tags assigned to the data points. This feature has the greatest impact on the classification process. We see both in Figure 8.3 and 8.4 that any combination of features which include the tag similarity reaches good results. We would anticipate a better performance for the combination of the tag and geographic location features. However, this is not the case. Therefore, we conclude that the use

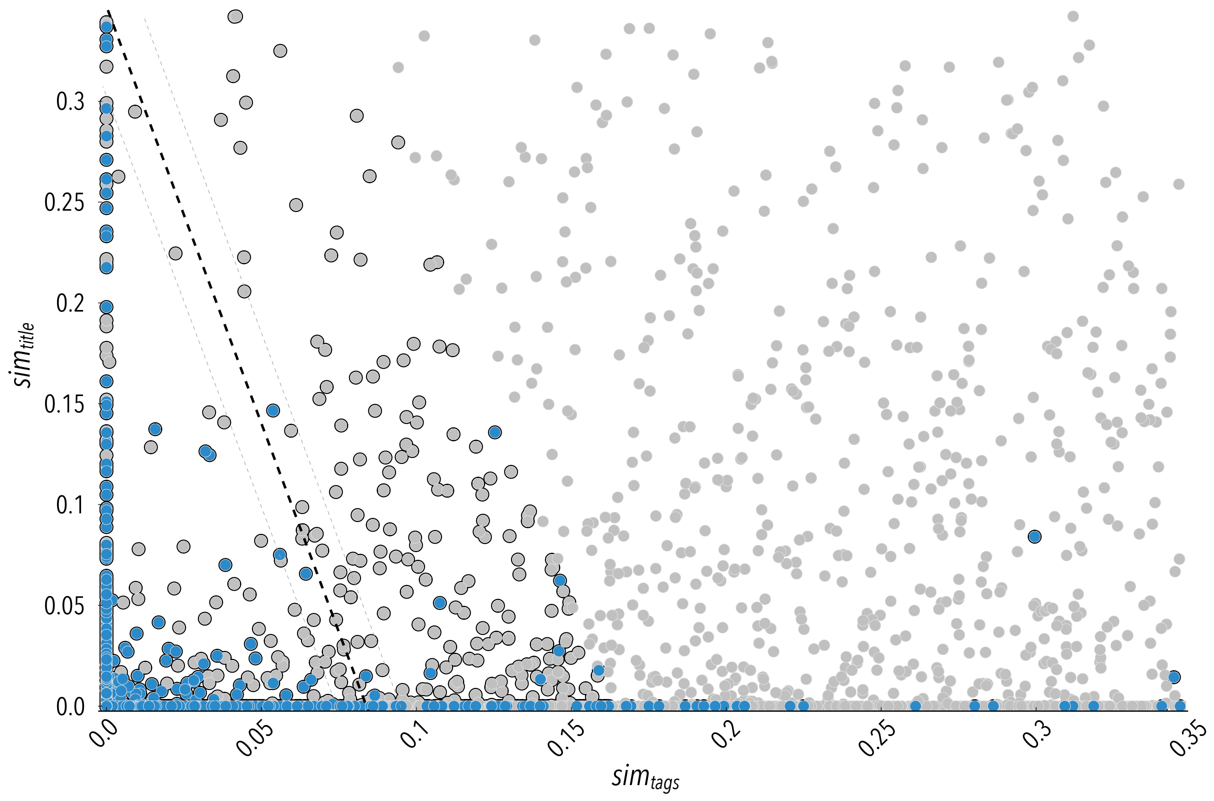


Figure 8.5.: Learned decision boundary of SVM using combination of sim_{tags} or $\text{sim}_{\text{title}}$

of the tag feature already is sufficient enough to correctly classify the documents in question where the geographic location would also be helpful for the decision. Overall, the final best performing settings use either a combination of sim_{tags} or sim_{title} (for the SVM) or sim_{tags} (for the Decision Tree) as features.

In Figure 8.5 we have graphically shown the problem to be solved by the SVM. It depicts the data points from the test set of the two classes as well as the decision boundary for the sim_{tags} and sim_{title} combination learned on the training set. We can see that a linear separation of the two classes is possible and that the number of misclassifications is acceptable. We also depicted the best performing Decision Tree using the sim_{tags} feature; this is shown in Figure 8.6. In this tree, the number of samples and the impurity is given for each node. As the impurity is based on the *Gini* impurity, it is denoted as *Gini* in the figure. We can see that the impurity for many leafs including many samples is low, thus showing that the decision model can decide well between the two classes.

Regarding the features for the new event detection we reach the same conclusion as regarding the similarity features. The results for the single features and the best performing combinations are shown in Table 8.4.

We can deduce from the heat maps and the Table 8.4 that the new event detection features

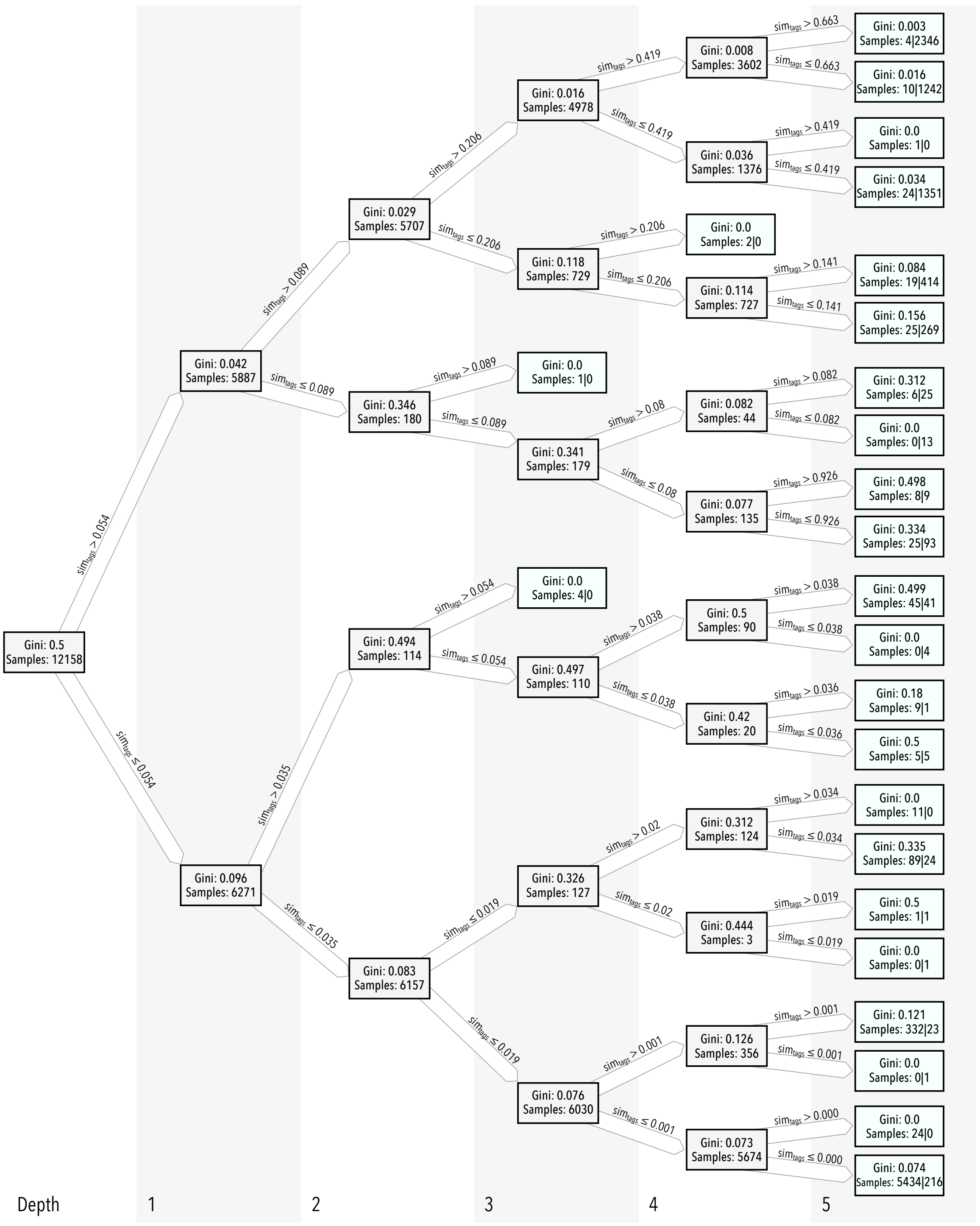


Figure 8.6.: Decision Tree of the similarity decision using tag feature only (best performing on overall task)

Table 8.4.: Best performance on overall task on training set using different new event detection features

New Event Detection Feature	SVM	Decision Tree
max	0.91	0.9089
min	0.8617	0.8619
avg	0.9034	0.8969
stddev	0.908	0.9045
max_{cap}	0.8614	0.8616
max_{upl}	0.8619	0.8624
$\text{max} + \text{avg} + \text{max}_{\text{upl}}$	**0.9101**	0.9087
$\text{max} + \text{min} + \text{max}_{\text{upl}}$	0.9097	**0.9094**
All	0.9098	0.9082

min, max_{cap}, and max_{upl} are not good features for the training of a classifier model for the correct prediction of whether to create a new event or not. These features show a very poor performance when used without the combination with other features. But even in conjunction with other better performing features, these features do not have a big impact on the overall performance. A look at the values in the heat maps shows that other features suit a lot better to solve the problem.

Even though a combination of three features (max, avg, and max_{upl} using a SVM and max, min, and max_{upl} using a Decision Tree) results in the best overall performance, the distance to the performance of the classifier trained only using the max feature is minimal. Therefore, we can question the usage of several features for this classification task into question and favor the usage of the single feature max over the combination of features.

Taking a look at the data points in Figure 8.5 shows that the usage of a linear kernel is sufficient to solve the problem. The efficiency of the linear kernel can also be seen in Figure 8.5. Experiments using both kernels show that the overall performance is comparable with either kernel used. The decision boundaries found using both kernels are almost identical.

Overall, our findings are very interesting as the number of features that are discriminative enough to ensure a further clustering is limited. Actually, the problems are solvable using one feature per problem. As a consequence, we doubt that the addition of another clustering pass can ameliorate the overall clustering further, unless other and new suitable features for measuring similarity are used.

8.4.3 Optimization of Candidate Retrieval Strategy

In Section 8.3.1 we have shown that the candidate retrieval strategies involving time features are the best working strategies. Taking a look at their efficiencies, the candidate retrieval strategy incorporating capture time is the most promising strategy. Therefore, we especially use this strategy for further optimization.

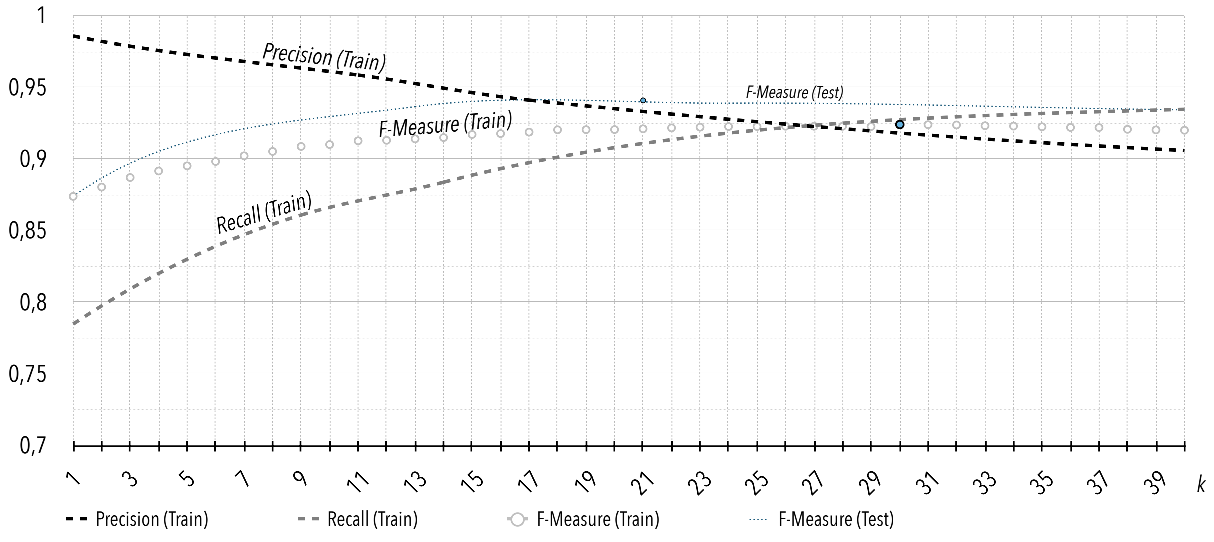


Figure 8.7.: Performance of the clustering framework in a two-pass paradigm using different numbers of candidates retrieved (using train and test set)

From our experiments in the previous section we have found the best settings for the other components using an exhaustive search. Therefore, we use these settings for the final optimization of the candidate retrieval step. For the scoring and ranking step we thus train a SVM classifier using the features *tags* and *title*. For the new event detection step we also use a Support Vector Machine. This classifier is trained with the best performing combination of features: max, avg, and max_{upl}. The best working threshold for the new event decision found for this combination is 0.18.

We employ our classification framework using the above mentioned settings. In a normal classification process, we let the system cluster the intermediate clustering produced by the *SimpleUserDayPreclustering* method. We do this process for a different number of candidates k to be retrieved by the candidate retrieval strategy. The results of these clustering processes are depicted in Figure 8.7 showing precision, recall, and F-measure for the overall clustering process with different numbers of candidates k. In addition to that, we show the F-measure for a clustering on the final data (test set) to be clustered using the same settings as for the training set.

We see that the highest value in terms of F-measure is reached when 30 candidates are retrieved ($k = 30$); the F-measure value reached is 92.4 %. This is exactly the number of candidates where the effectiveness for this candidate retrieval strategy has reached its maximal slope (see Figure 8.2).

It can be observed that the obtained value for k is not the optimal value when applied to the final test set. However, the performance does not vary too much if a certain amount of candidates is retrieved. Therefore, this difference of performance is acceptable.

Table 8.5.: Results for clustering using different strategies for the first pass and an optimized second pass

Method	F_1-Measure	Precision	Recall	Time Consumption
Our single-pass approach	0.886	0.951	0.83	1.000 (Baseline)
NormalPreclustering	0.932	0.92	0.945	1.102 (+10.2 %)
NormalPreclusteringByDay	0.931	0.938	0.925	0.744 (−25.5 %)
SimpleUserDayPreclustering	**0.939**	0.935	0.943	**0.009 (−90.1 %)**

8.5 Results of the Clustering Framework used in Two-Pass Mode

In this section, we compare the overall system performance of our framework in a multi-pass setting. We have constructed the system to use a two-pass strategy and have chosen an overall system setting where the second pass uses a fixed strategy in combination with different first-pass strategies. The approach which is employed as second pass is described in more detail in the previous sections. The first-pass strategies comprise one very costly as well as one very cheap strategy. In the following we show the overall performance of all these combinations which are measured in terms of precision, recall, and F-measure. In addition to that, we measure the time consumption to get an idea for its computational effectiveness.

We present the combination of the second-pass strategy with the following three first-pass strategies: *NormalPreclustering*, *NormalPreclusteringByDay*, and *SimpleUserDayPreclustering*. As we have already stated in Section 8.1, the time consumed by our single-pass approach is used as a reference (we set the time needed to 1.0). If a two-pass system consumes more computational time, the value will thus be greater than 1.0. For the time consumption of the first-pass strategies see Table 8.1.

The final results of all approaches are shown in Table 8.5. All experiments have been conducted on the test set of our dataset. Therefore, all results are comparable with those presented using the single-pass strategy as well as other approaches relying on this dataset for clustering.

As we can see in the table, the results are very surprising. The key observations of these results are the following:

- Our initial intention to increase the recall using a second pass was successful. The addition of that pass increases the overall performance of the system significantly, so that our best performing strategy reaches almost 94 % in terms of F-measure. This is more than 5 percentage points better than the best performing single-pass strategy (93.9 % vs. 88.6 %). Independent of the first-pass strategy the increase of the overall results in terms of F-measure is very high and the results are appealing.
- If the intermediate clustering produced in the first pass is pure (has a high precision), the system reaches a higher overall performance when using a two-pass strategy while keeping the precision at a high level. In comparison to the single strategy we only lose about 1 percentage point in terms of precision. We assume that this high precision is still acceptable from an end user perspective.

- A first-pass strategy with a high precision is preferred over strategies not having such a high precision. The *SimpleUserDayPreclustering* provides an intermediate clustering with very high precision. In comparison to the *NormalPreclustering* strategy we see that the precision of the final clustering is also higher. We can conclude that an intermediate clustering having high precision is an optimal starting point for a two-pass strategy.

- Regarding the scalability of the overall system, a two-pass strategy is more effective than a single-pass strategy. We could decrease the time consumption by one order of magnitude in comparison to the best performing single-pass strategy. This is remarkable as this is also the best performing two-pass strategy.

We are aware that we reach a higher performance in comparison to the runs on the training set. We would expect this to be the opposite. This phenomenon has already been observed by other researchers using this dataset during the MediaEval 2013 Social Event Detection challenge. Indeed, the assembly of the test set seems to be easier to cluster than the training set.

8.6 Conclusions

In this chapter we have presented a multi-pass approach for clustering with our event-based stream classification framework. We have shown that we are indeed able to classify a stream of social media data into a growing and evolving set of events using a multi-pass strategy. The methods used have been evaluated on our dataset. Therefore, the results can be easily compared with other approaches working on this publicly available dataset.

Before starting the experiments with the multi-pass approach, we identified several key problems which we can successfully address:

- We are able to tackle the overall clustering problem and achieve a better performance in comparison with the single-pass approach, i. e. we reach an F-measure of about 93.9 % instead of 88.6 %.

- We are still able to scale to the data sizes and data rates which we encounter in social media applications. We can even increase the computational efficiency by one order of magnitude by using two appropriate classification passes.

- We are still able to stick to a stream paradigm which is needed to process data from social media applications. We have found a first-pass strategy that allows to hand over the data in a preprocessed way so that the general idea of the stream clustering stays intact. Nevertheless, this has the cost of a short delay.

All these problem were addressed successfully by using a two-pass strategy where the first pass uses a simple strategy of merging the data items by user and day. The second pass is then a more thorough classification process using machine learning techniques.

We have compared different intermediate clusterings showing that an intermediate clustering with high precision is a good starting point for further processing in a second pass. All approaches using the two-pass paradigm significantly outperform all approaches using one pass while the processing time is even lower. The overall performance of almost 94 % F-measure is very good. From an end user perspective and regarding the purity of the event clusters, the clustering is also of very good quality. It is difficult to find better features to improve the clustering furthermore.

PART IV

Concluding Remarks

CHAPTER 9

Remarks and Comparison of Clustering Approaches

In the first three parts of this dissertation we have presented our classification framework both for single-pass and multi-pass settings. We have seen that both approaches are valid and effective methods that can successfully address the challenge of discovering events and identifying which documents are associated with the same event in a data stream of social media documents. That is, both approaches can be used to cluster a real-world dataset comprised of social media documents, such as images from Flickr together with their associated metadata.

In this chapter we provide additional remarks on our contribution. We will discuss prerequisites for our approaches presented in Part II and III. Furthermore, we talk about their advantages and disadvantages and discuss how usable these approaches are in a real-world setting.

After that, we take a look at several approaches which we already discussed as related work. We especially consider the approaches which used our dataset and reported results on its test set. We provide figures and perform a direct comparison between them and our system.

9.1 Prerequisites for Event Clustering

In order to cluster data from social media applications, we require some preparatory work before our classification framework can be applied. More specifically, in order to cluster social media data, the data itself has to be preprocessed so that only documents are passed to our classification framework that represent an event. As a consequence, we need another step in addition to our processing pipeline which is capable of filtering out all data points that do not represent events.

In our scenario we have chosen a starting point where the data has already been filtered by using data that has been constructed using an online event calendar. Even though this is an optimal way to create, train, and evaluate an event detection system like our approach, such type of data is not available when using raw data directly from a social media application.

In order to preprocess raw data and produce suitable data for event detection, another approach using machine learning techniques can provide a filtering by deciding whether a document depicts an event or not. Such so-called event identification systems already exist (e. g. Chen and Roy [CR09] and Liu, Troncy, and Huet [LTH11]). Therefore, an event identification system has to be employed as an initial step before the resulting outcome can be fed to our classification framework. In our scenario, one requirement for such a process is that it can deal with a stream of data, that is the decision has to be made immediately after the data point has arrived.

An alternative solution is to integrate the event/no-event decision directly into our framework. In that case, a preprocessing of the data is not necessary. This can be done by enhancing the new event detection step. For this step we can introduce a different classifier that supports a classifier into more than two classes, thus using a multi-class instead of a binary classifier. As a result, we do not only use the new event detection step to decide whether the current document depicts a new event or not, but we also decide if the data point represents an event at all. Therefore, we end up with three target classes for the decision.

It is difficult to estimate how well these strategies perform; further research has to investigate this issue. Another important challenge is to find possibilities to create appropriate data including a gold standard making training of a suitable decision model possible.

9.2 Reflection on Multi-Pass Clustering in a Stream-based Setting

The classification of a never-ending stream of documents into classes in real time has certain prerequisites in order to be feasible. One of these prerequisites is the limitation of the possible number of passes. In this section we aim to analyze this further and reflect on the consequences regarding our multi-pass approach.

Overall, we can assume that the following four challenges need to be addressed in order to identify events in data from social media applications:

1. Approaches need to tackle the social media documents in a stream-based fashion. The order of the data points is thus not changeable and each incoming data point has to be processed immediately.
2. Approaches need to address the fact that the data stream is never-ending and of large scale.
3. Approaches need to determine the number of target event classes which is unknown during the whole clustering process.
4. Approaches need to do the classification in real time so that the clustering is available for further processing (e. g. displaying) immediately.

Not all classification approaches meet all of these challenges. In the following we take a look at the classification approaches presented in this work. The challenges 1 to 3 are fully addressed by all of them, i. e. the problems described can be tackled by all our approaches regardless whether our single-pass or multi-pass approach is used. When using our single-pass strategy, challenge 4 is also fulfilled without restrictions. However, we confess that—strictly speaking—challenge 4 is fulfilled only partially by the approaches using more than one single pass. Actually, when using our two-pass approach, the final event clustering is not made available in real time but has a delay of up to 24 hours. This time delay can be shortened if the first clustering pass uses smaller windows, but, in general, a delay cannot be avoided. This is problematic for applications where an immediate clustering is important.

As a consequence, if all of the above mentioned points have to be fully met, the classification process is indeed limited to one single pass. In that case we are faced with a dilemma to decide whether we need a finalized clustering in real time with the cost of a lower performance or if a delay of some duration is acceptable. As both solutions are not fully satisfying, we want to propose a solution to that issue. We propose the following strategies to tackle the problem of fulfilling the real-time requirement in a multi-pass approach:

- An application initially invokes our single-pass strategy to achieve an initial clustering. This initial clustering is used for the presentation of event clusters to the end user for the most recent documents. Event clusters that are older than a certain time are then classified again in a second pass refining the clustering. For this we propose to use our second-pass approach as proposed in our two-pass setting. The application therefore is able to show all event clusters immediately, where events that are older than the current day have a higher quality than the clusters of the current day.
- Instead of using our single-pass strategy, the per day and user merging strategy is used as an initial clustering. The application uses a clustering per user for the current day for displaying. The second pass is used afterwards refining the clustering as proposed before, reaching a significantly higher quality for days older than the current one.

Both approaches allow to present an immediate clustering but have different strengths. When using the first strategy, the initial clustering is of higher quality than the second strategy (88.6 % vs. 84.2 % F-measure). In contrast, the second strategy has its advantages in the long term. The overall clustering quality for the older documents and event clusters is a bit higher in terms of F-measure: 93.9 % vs. 93.2 %. But more importantly, the precision is also higher using the second strategy and therefore predicted to be better accepted from an end user's perspective (93,5 % vs. 92.0 %). Therefore, it depends on the need of the application to decide which strategy suits better. If a good clustering is needed for documents that appeared recently, then the first strategy is the preferred one. For an overall better quality, the second strategy is the preferred one.

We can also think of a third strategy which combines the advantages of both approaches. The best performing single-pass and the best performing multi-pass approaches are applied at the same time. The event clusters with the data points from the current day are produced by our single-pass strategy. On the other hand, the older event clusters are produced by the multi-pass strategy. An application then uses the clustering from the single-pass strategy for the documents of the current day and the multi-pass strategy for less recent documents. Using this strategy, it is important to remark that it comes at the expense of the need of more computational time.

Overall, we see that a multi-pass approach and a real-time classification are not necessarily contradictions, rather the challenge can be tackled using one of the strategies proposed above.

9.3 Comparison with Other Approaches

In the results section of our approaches we only compared them to other work that can be used in a stream-based setting. Nevertheless, there are several other approaches which not necessarily use a stream-based setting but used our ReSEED dataset and provide results on a clustering for the test set. In the following we compare our approach to their results. These approaches have already been described in more detail earlier in this thesis (please refer to Section 3.7).

The results in terms of F-measure for a clustering on the test set of our dataset are shown in Table 9.1. We only provide the F-measure values without precision and recall as these figures were not reported in the original publications.

Table 9.1.: Comparison of different clustering strategies of MediaEval 2013 SED with our classification approaches

Reference	Algorithm/Method	F-measure
[GGC13]	Location-Time Merging	0.143
[Pap+13b]	Chinese Restaurant Process	0.236
[Raf+13]	Hierarchical-like Merging	0.570
[Sch+13]	Graph Clustering	0.704
[ZZD13]	Meanshift Clustering + LDA	0.740
[BI13]	SVM Classification	0.780
[SN13]	Constrained Spherical K-Means	0.812
[WS13]	Quality Threshold Clustering	0.878
[MG13]	Photo-TOC	0.883
Chapter 6	**Single-Pass SVM Classification**	**0.886**
[Ngu+13]	User-centric Split and Merge Clustering	0.932
Chapter 8	**Multi-Pass SVM Classification**	**0.939**
[Sam+13]	DBSCAN Clustering	0.946

The table contains the approaches presented at the Social Event Detection task organized as part of MediaEval 2013 together with our two best-performing approaches. We see that there are many approaches that can be used to produce a clustering on the dataset. However, almost all of these approaches are based on classical clustering algorithms that cannot be used in a stream-based setting. Nevertheless, the figures in the table give an overview of how well the different approaches perform in comparison to our single- and multi-pass approach.

There is only one approach which is able to outperform our multi-pass approach in terms of F-measure. This approach from Samangooei et al. [Sam+13] uses *DBSCAN*, a density-based clustering algorithm. Together with the solution from Nguyen et al. [Ngu+13] and our multi-pass strategy, these are the only approaches that reach a very good overall performance with more than 90 % in terms of F-measure. Even though our system only outperforms one of them, our overall results are competitive, as the difference between the results is minimal (difference of 0.7 percentage points). But it is not neglectable that our classification-based approach has several advantages (like working in a stream-based scenario) that may be more important than the minimal gain in overall performance.

The next best approach which can be used in a stream-based setting is the one from Brenner and Izquierdo [BI13] who also employ an SVM for classification. However, this approach only reaches an F-measure of about 78 %, which is clearly outperformed by both of our strategies. It is also remarkable that our single-pass strategy outperforms many approaches using clustering algorithms. This is interesting as our approach only needs one single pass, thus sees the data points only once. In addition, it is very effective regarding the computational effort needed. We know from at least some of the other approaches that they are computationally challenging and do not scale well to larger datasets.

Overall, this comparison clearly shows that our approaches are able to get the most out of the data while having several advantages over other approaches regarding the efficiency and way they handle the data. They are able to outperform several state-of-the-art approaches and are promising for deployment in an actual real-world application.

CHAPTER 10

Conclusion

In this thesis we have been concerned with the problem of event detection in social media. We have addressed the challenge of classifying a massive, never-ending stream of data from a social media application into corresponding events. This clustering task involved several challenges. As a solution we have presented a framework tackling this task together with its challenges. We were able to show that our event-based stream classification framework is capable of classifying the social media data into a growing and evolving set of events. Furthermore, it successfully addresses all challenges identified before. Overall, we provide several contributions which we want to outline in the following.

We introduced a new dataset to support research in the area of social event detection. The dataset was created using images from the Flickr photo community site and online event calendars. Therefore, it contains user-contributed images together with associated metadata describing the events they depict. All data included in the dataset is licensed under a Creative Commons license that allows for free distribution. The dataset was previously published and is freely available to the public. Both the scale and complexity of the dataset make it more challenging and more representative of real-world problems in comparison to other datasets in the field, which are usually annotated explicitly and are much smaller. We see this dataset as an important contribution to the advancement of the social event detection field. It supports the development, evaluation, as well as the systematic comparison of different social event detection approaches. Furthermore, it can be used for clustering and classification tasks in that area. As it has been already used with the MediaEval Social Event Detection task in 2013, it is also known in the community and facilitates the comparison of different approaches.

Having created a suitable dataset, we introduced a classification framework that is able to cluster the documents into their appropriate event cluster using different machine learning techniques. The system we have presented is able to classify a stream of social media data in one single pass. In particular, we have shown that our approach successfully addresses several key problems:

- Handling the data in a stream-based paradigm, i. e. tackling the problem of a stream of data that is never-ending.
- Scaling to the data sizes as well as data rates that are usually encountered in social media applications.

- Tackling the new event detection problem, i. e. the problem of determining whether an incoming data point belongs to a new or to an already known event.

We successfully address these problems i) by employing a classification algorithm, therefore allowing us to process the data in one pass, ii) by including a candidate event retrieval step which retrieves a set of event candidates that the incoming data point is likely to belong to, and iii) by including a function trained using machine learning techniques that determines whether the incoming data point belongs to the top-scored candidate or rather to a new event.

To maximize the performance of our classification system, we have examined all different steps of our classification framework thoroughly. In particular, we have shown that the use of a suitable candidate retrieval step is crucial to reduce the number of events that are considered as potential candidates to which the incoming data point could belong to. This is necessary to scale up to the data sizes an data rates found in social media applications. Therefore, we have not only examined and experimentally compared different candidate retrieval strategies regarding their effectiveness and computational cost but also regarding their performance on the overall classification task. A candidate retrieval strategy based on the timestamps selecting the 18 closest events with respect to upload time delivers the best results. From a more general perspective, the results found show that a suitable candidate retrieval strategy indeed does not only have the potential to speed up the classification process per se, but also helps to filter out false positives that could potentially confound the classifier. Thus, this step enables us to increase the overall classification performance.

Furthermore, we have shown that both the decision finding the top scoring target event as well as the new event detection decision can be taken with reasonable accuracy. For both decisions we made use of different types of classifiers. We used different types of Support Vector Machines (Standard and Ranking SVM) as well as Decision Trees in order to train appropriate models. Furthermore, we investigated different strategies for creating appropriate training data, settings for the classifiers, and the impact of the amount of training data on the performance.

We have shown that it is possible to learn a suitable similarity measure using all three types of classifiers. Using an SVM, only a few training examples are sufficient, already allowing the induction of an acceptable similarity measure with less than 20 training examples. The performance of an SVM does not vary too much for different sizes of training data and the number of training data needed is much less than with a Decision Tree. Thus we can regard an SVM as more robust than to a Decision Tree. Besides, using an SVM we figured out that a linear kernel performs as good as a RBF kernel regardless of which type of SVM is used, making the use of a linear kernel the favored strategy mainly because it is computationally less expensive. It has also been shown that the sampling strategy is crucial for the maximization of the success of the training process. The use of well-chosen training examples improves the quality of the created model. We have seen that the search for the nearest wrong pair helps to train a good model.

For the new event decision model, we have seen that using 4,000 training examples together with an SVM maximizes the decision performance. Here, the number of training examples chosen is less crucial than in the creation of the similarity measure and the performance reached by the different models is more robust.

We carried out detailed feature analyses for both decision problems. The aim in both cases was to find an optimal combination of features to ensure the creation of an optimized decision model. Many pictures in social media are missing some of the features as the user did not provide them. But in the feature analysis for the learning of a similarity measure, we have discovered that the lack of a single feature has no big impact on the quality of the system as a whole. The results are still compelling even if only one feature is used in addition to the always available time features. Therefore, our method is still applicable, as we can deduce from our dataset that contains typical data of a social media application that about 99.8 % of the included documents have at least one additional feature assigned in addition to the time features. We thus could clearly show that we can deal with the lack of data which usually occurs in real-world scenarios.

Our second feature analysis, carried out for the identification of the optimal combination of features to be used for the new event detection decision, was a greedy strategy. We identified that a combination of four out of six features—originally derived as a result from an optimal clustering—yields the best performance. We also showed that a model using these features outperforms a simple threshold model using only the maximum value of the top-scoring event.

The optimized system has been compared to several baselines showing that it outperforms all of them in terms of F-measure with a value of 88.6 %. The benefit of our approach is that the processing time per document remains nearly constant with an increasing number of documents, thus addressing the scalability challenge mentioned above. We found that the resulting clustering was very pure (i. e. having a high precision) while having a lower recall. Regarding the results, the question arose whether the recall can still be increased by using a multi-pass strategy.

We therefore introduced an extension of the single-pass approach so that it could be used in a multi-pass setting. We finally showed that the extended approach can increase the recall significantly when using two passes. This also increased the overall results, now being significantly better. In addition to that, we could also lower the computational time needed for the processing by 90 % using the best performing two-pass strategy.

Creating the multi-pass system we analyzed several strategies in order to find a suitable strategy to be used for each pass. We experimented with different first-pass strategies showing that all two-pass strategies clearly outperform the single-pass strategy. We saw that a simple merging by day and user used as a cheap first pass clustering—taking no more than two seconds for the whole test set on a recent computer system—performs best in conjunction with our classification framework as a second pass for the refinement of the intermediate clustering. This combination reached an F-measure of almost 94 %.

We then looked at the single steps of our framework and re-optimized all needed steps of our classification framework to the new situation using an intermediate clustering for further merging. We found that the number of features needed for the similarity measure as well as the new event detection decision almost shrunk to one single feature. Using the cosine similarity of the tag feature performs best for measuring similarity of an intermediate event cluster to a final event cluster. The new event detection decision can be done using the maximum value of the top-scoring event candidate. The optimization of the candidate retrieval step showed that a strategy similar to the one used in the single-pass setting works well. The difference is that the number of candidates has to be increased from 18 to 30 candidates to be retrieved with respect to the capture time.

Summarizing, we presented an event-based stream classification framework that can be used either in a single- or a multi-pass setting. We showed that this framework is capable of successfully clustering a real-world and non-toy dataset. Our highly efficient single-pass strategy reached an F-measure of 88.6 %, already outperforming several state-of-the-art approaches. Our two-pass strategy reached an F-measure of 93.9 %, ameliorating the overall performance significantly. In addition, the computational effort needed has been lowered by one order of magnitude in comparison to our single-pass strategy. In a comparison to other approaches applied to our dataset, we have shown that our approach is one of the best performing ones. We also concluded that our approach can indeed be used for a real-world application.

Outlook

We see possibilities to enhance our event-based stream classification framework to tackle the shortcoming that it can only use data that has been filtered before, so that only documents are fed that clearly constitute an event. We already proposed to change the new event detection step so that it can also decide whether the document depicts an event or not. Approaches from Chen and Roy [CR09] and Liu, Troncy, and Huet [LTH11] show that the problem can be tackled with machine learning techniques. Therefore, we think that such an extension is possible and will make the framework more versatile.

Moreover, it would be interesting to apply our framework to other datasets derived from other social media applications. Our system could contribute to classifying events found on Twitter, for example. Another very interesting application is to combine data from different social media application sources. We are sure that our classification framework can help in finding links between events that are spread over different social media channels, thus making a use of data across sites possible in order to create a full event story.

However, the scenario presented in this work does not reflect the natural event constitution which changes over time. It is therefore desirable to achieve a hierarchy of events so that the end-user can explore the events through such a hierarchy. Using our multi-pass strategy together with a suitable dataset which models a hierarchy can be used to identify event hierarchies and make the exploration even more natural for the end user, e. g. by showing more sub-events for recent activities and switch to a more coarse view for older events.

PART V

Appendix

Glossary

API An Application Programming Interface (API) is a set of routines, protocols, and/or tools which can be used to build applications using a third party service. They are independent functionalities which are independent from the actual implementation of the third party application. 50, 51, 53, 54, 56, 57

DARPA The Defense Advanced Research Projects Agency (DARPA) is an agency of the United States Department of Defense responsible for the development of new technologies for use by the military. 25

EXIF The exchangeable image file format (Exif) is a standard which specifies the format for image information used by digital cameras to add image information to an image. 57, 59

Facebook Facebook is an online social networking service, a platform to build social networks among people that share interests and want to connect with each other. 3, 61

Flickr Flickr is an online photo community by Yahoo Inc. It is an image hosting web service enabling users to share personal photographs which can be enriched with metadata like keywords. 3, 49–52, 54–59, 61, 67

GPS The Global Positioning System (GPS) is a satellite-based navigation system providing location and time information anywhere on the earth. 57, 76, 98

HTTP The HyperText Transfer Protocol (HTTP) is an application protocol to exchange or transfer hypertext. It is structured hypertext which uses links between single nodes containing text. 50, 52

JPEG JPEG is a commonly used file format with a lossy compression method that is used for the storage of digital images. 59

MediaEval MediaEval is a benchmarking initiative dedicated to evaluating new algorithms for multimedia access and retrieval. 27

tweet A tweet is a message with 140 characters that is sent using the Twitter service. 6, 27

Twitter Twitter is an online social networking service. It allows users to send 140-character messages called tweets that others can follow. 3

YouTube YouTube is a video-sharing web site which allows users to upload, view, and share videos. It also has the possible to annotate videos. 3

Acronyms

ACE	Automatic Content Extraction. 26
DARPA	Defense Advanced Research Projects Agency. 25, *Glossary:* DARPA
LREC	Language Resources and Evaluation Conference. 26
MED	Multimedia Event Detection. 26, 49
MediaEval	MediaEval Benchmark for Multimedia Evaluation. 27, 49, 61, *Glossary:* MediaEval
SED	Social Event Detection. 27, 49
TDT	topic detection and tracking. 25, 27
TRECVID	TREC Video Retrieval Evaluation. 26, 49

Bibliography

[Agg07] Charu C. Aggarwal. *Data Streams: Models and Algorithms*. Vol. 31. Springer, 2007.

[Agg+04] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. "On Demand Classification of Data Streams". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2004, pp. 503–508.

[AY05] Charu C. Aggarwal and Philip S. Yu. "An effective and efficient algorithm for high-dimensional outlier detection". In: *The VLDB journal* 14.2 (2005), pp. 211–221.

[AY01] Charu C. Aggarwal and Philip S. Yu. "Outlier Detection for High Dimensional Data". In: *ACM Sigmod Record*. Vol. 30. 2. ACM. 2001, pp. 37–46.

[AY08] Charu C. Aggarwal and Philip S. Yu. "Outlier Detection with Uncertain Data". In: *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, 2008, pp. 483–493.

[All02] James Allan. *Topic Detection and Tracking – Event-based Information Organization*. Springer, 2002.

[All+98] James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. "Topic Detection and Tracking Pilot Study Final Report". In: *The DARPA Broadcast News Transcription and Understanding Workshop*. 1998.

[APL98] James Allan, Ron Papka, and Victor Lavrenko. "On-line New Event Detection and Tracking". In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '98. Melbourne, Australia: ACM, 1998, pp. 37–45. ISBN: 1-58113-015-5. DOI: 10.1145/290941.290954.

[Bab+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. "Models and Issues in Data Stream Systems". In: *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '02. Madison, Wisconsin: ACM, 2002, pp. 1–16. ISBN: 1-58113-507-6.

[BDM04] Brian Babcock, Mayur Datar, and Rajeev Motwani. "Load Shedding for Aggregation Queries over Data Streams". In: *Proceedings. 20th International Conference on Data Engineering, 2004*. IEEE. 2004, pp. 350–361.

[BR93] Jeffrey Banfield and Adrian Raftery. "Model-based Gaussian and Non-Gaussian Clustering". In: *Biometrics* (1993), pp. 803–821.

[BLC02] Daniel Barbará, Yi Li, and Julia Couto. “COOLCAT: An Entropy-based Algorithm for Categorical Clustering”. In: *Proceedings of the Eleventh International Conference on Information and Knowledge Management*. ACM. 2002, pp. 582–589.

[Bar+03] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. “Learning Distance Functions Using Equivalence Relations”. In: *ICML*. Vol. 3. 2003, pp. 11–18.

[Bar63] Roger G. Barker. “The Stream of Behavior as an Empirical Problem.” In: *The Stream of Behavior*. Ed. by Roger G. Barker. Appleton-Century-Crofts, 1963, pp. 1–22.

[BF01] Kobus Barnard and David A. Forsyth. “Learning the Semantics of Words and Pictures”. In: *ICCV*. 2001, pp. 408–415.

[Bar88] Lawrence W. Barsalou. “The Content and Organization of Autobiographical Memories”. In: *Remembering Reconsidered: Ecological and Traditional Approaches to the Study of Memory* (1988), pp. 193–243.

[BBM02] Sugato Basu, Arindam Banerjee, and Raymond Mooney. “Semi-supervised Clustering by Seeding”. In: *In Proceedings of 19th International Conference on Machine Learning (ICML-2002)*. Citeseer. 2002.

[BBM04a] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. “Active Semi-Supervision for Pairwise Constrained Clustering”. In: *SDM*. Vol. 4. SIAM. 2004, pp. 333–344.

[BBM04b] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. “A Probabilistic Framework for Semi-supervised Clustering”. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’04. Seattle, WA, USA: ACM, 2004, pp. 59–68. ISBN: 1-58113-888-1. DOI: 10.1145/1014052.1014062.

[BCC03] Rohan Baxter, Peter Christen, and Tim Churches. “A Comparison of Fast Blocking Methods for Record Linkage”. In: *ACM SIGKDD*. Vol. 3. Citeseer. 2003, pp. 25–27.

[Bec+12] Hila Becker, Dan Iter, Mor Naaman, and Luis Gravano. “Identifying Content for Planned Events Across Social Media Sites”. In: *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*. WSDM ’12. Seattle, Washington, USA: ACM, 2012, pp. 533–542. ISBN: 978-1-4503-0747-5. DOI: 10.1145/2124295.2124360.

[BNG09] Hila Becker, Mor Naaman, and Luis Gravano. “Event Identification in Social Media.” In: *WebDB*. 2009.

[BNG10] Hila Becker, Mor Naaman, and Luis Gravano. “Learning Similarity Metrics for Event Identification in Social Media”. In: *Proceedings of the Third ACM International Conference on Web Search and Data Mining*. ACM. 2010, pp. 291–300.

[BY01] Gill Bejerano and Golan Yona. “Variations on Probabilistic Suffix Trees: Statistical Modeling and Prediction of Protein Families”. In: *Bioinformatics* 17.1 (2001), pp. 23–43.

[Ben02] Jonathan Bennett. “What Events Are”. In: *The Blackwell Guide to Metaphysics.* Ed. by Richard M. Gale. Blackwell, 2002, pp. 43–65.

[B+99] Kristin Bennett, Ayhan Demiriz, et al. “Semi-supervised Support Vector Machines”. In: *Advances in Neural Information processing systems* (1999), pp. 368–374.

[Ben+96] Amine M. Bensaid, Lawrence O. Hall, James C. Bezdek, and Laurence P. Clarke. “Partially Supervised Clustering for Image Segmentation”. In: *Pattern Recognition* 29.5 (1996), pp. 859–871.

[BBM04c] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. “Integrating Constraints and Metric Learning in Semi-supervised Clustering”. In: *Proceedings of the Twenty-first International Conference on Machine Learning.* ICML ’04. Banff, Alberta, Canada: ACM, 2004, pp. 11–. ISBN: 1-58113-838-5. DOI: 10.1145/1015330.1015360.

[BM03] Mikhail Bilenko and Raymond J. Mooney. “Adaptive Duplicate Detection Using Learnable String Similarity Measures”. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* Wachington, DC, 2003.

[Bol92] Marilyn G. Boltz. “The Remembering of Auditory Event Durations”. In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 18.5 (1992), p. 938.

[B+98] Paul S. Bradley, Usama M. Fayyad, Cory Reina, et al. “Scaling Clustering Algorithms to Large Databases”. In: *KDD.* 1998, pp. 9–15.

[Bre+84] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and Regression Trees.* CRC press, 1984.

[BI13] Markus Brenner and Ebroul Izquierdo. “MediaEval 2013: Social Event Detection, Retrieval and Classification in Collaborative Photo Collections”. In: *MediaEval 2013 Workshop.* Spain, 2013.

[Bre+00] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. “LOF: Identifying Density-based Local Outliers”. In: *SIGMOD Rec.* 29.2 (May 2000), pp. 93–104. ISSN: 0163-5808. DOI: 10.1145/335191.335388.

[BS98] Norman R. Brown and Donald Schopflocher. “Event Clusters: An Organization of Personal Events in Autobiographical Memory”. In: *Psychological Science* 9.6 (1998), pp. 470–475.

[Bud+06] Suratna Budalakoti, Ashok N. Srivastava, Ram Akella, and Eugene Turkov. *Anomaly Detection in Large Sets of High-dimensional Symbol Sequences.* Tech. rep. NASA ARC, 2006.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly Detection: A Survey”. In: *ACM Comput. Surv.* 41.3 (July 2009), 15:1–15:58. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882.

[CL11] Chih-Chung Chang and Chih-Jen Lin. “LIBSVM: A Library for Support Vector Machines”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3 (2011), p. 27.

[C+06] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. *Semi-supervised Learning.* Vol. 2. MIT press Cambridge, 2006.

[CR09] Ling Chen and Abhishek Roy. “Event Detection from Flickr Data through Wavelet-based Spatial Analysis”. In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management.* ACM. 2009, pp. 523–532.

[CM07] Vladimir Cherkassky and Filip M. Mulier. *Learning from Data: Concepts, Theory, and Methods.* John Wiley & Sons, 2007.

[CCM03] David Cohn, Rich Caruana, and Andrew McCallum. “Semi-supervised Clustering with User Feedback”. In: *Constrained Clustering: Advances in Algorithms, Theory, and Applications* 4.1 (2003), pp. 17–32.

[Con05a] Linguistic Data Consortium. *ACE (Automatic Content Extraction) English Annotion Guidelines for Events.* 2005. URL: `https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf`.

[Con05b] Martin A. Conway. “Memory and the Self”. In: *Journal of Memory and Language* 53.4 (2005), pp. 594–628.

[CP00] Martin A. Conway and Christopher W. Pleydell-Pearce. “The Construction of Autobiographical Memories in the Self-memory System.” In: *Psychological Review* 107.2 (2000), p. 261.

[Coo+05] Matthew Cooper, Jonathan Foote, Andreas Girgensohn, and Lynn Wilcox. “Temporal Event Clustering for Digital Photo Collections”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)* 1.3 (2005), pp. 269–288.

[CV95] Corinna Cortes and Vladimir Vapnik. “Support-Vector Networks”. In: *Machine Learning* 20.3 (1995), pp. 273–297.

[Dan63] Arthur Danto. “What We Can Do”. In: *The Journal of Philosophy* 60.15 (1963), pp. 435–445.

[Dav80] Donald Davidson. *Essays on Actions and Events.* New York: Oxford University Press, 1980.

[Dav69] Donald Davidson. *The Individuation of Events.* Springer, 1969.

[Dav+07] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. “Information-theoretic Metric Learning”. In: *Proceedings of the 24th International Conference on Machine Learning.* ICML ’07. Corvalis, Oregon: ACM, 2007, pp. 209–216. ISBN: 978-1-59593-793-3. DOI: `10.1145/1273496.1273523`.

[Dav+06] Manuel Davy, Frédéric Desobry, Arthur Gretton, and Christian Doncarli. “An Online Support Vector Machine for Abnormal Events Detection”. In: *Signal Processing* 86.8 (2006), pp. 2009–2025.

[DMC03] Tijl De Bie, Michinari Momma, and Nello Cristianini. “Efficiently Learning the Metric with Side-Information”. In: *Algorithmic Learning Theory*. Ed. by Ricard Gavaldá, Klaus P. Jantke, and Eiji Takimoto. Vol. 2842. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 175–189. ISBN: 978-3-540-20291-2. DOI: 10.1007/978-3-540-39624-6_15.

[DBE99] Ayhan Demiriz, Kristin P. Bennett, and Mark J. Embrechts. “Semi-Supervised Clustering Using Genetic Algorithms”. In: *In Artificial Neural Networks in Engineering*. ASME Press, 1999, pp. 809–814.

[Dod+04] George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie Strassel, and Ralph M. Weischedel. “The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation.” In: *LREC*. 2004.

[Don+03] Guozhu Dong, Jiawei Han, Laks V. S. Lakshmanan, Jian Pei, Haixun Wang, and Philip S. Yu. “Online Mining of Changes from Data Streams: Research Problems and Preliminary Results”. In: *Proceedings of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams*. 2003.

[Eic+05] Christoph F. Eick, Alain Rouhana, Abraham Bagherjeiran, and Ricardo Vilalta. “Using Clustering to Learn Distance Functions for Supervised Similarity Assessment.” In: *MLDM*. Ed. by Petra Perner and Atsushi Imiya. Vol. 3587. Lecture Notes in Computer Science. Springer, 2005, pp. 120–131. ISBN: 3-540-26923-1.

[EZZ04] Christoph F. Eick, Nidal Zeidat, and Zhenghong Zhao. “Supervised Clustering – Algorithms and Benefits”. In: *ICTAI 2004. 16th IEEE International Conference on Tools with Artificial Intelligence, 2004*. IEEE. 2004, pp. 774–776.

[EVE02] Mohamed G. Elfeky, Vassilios S. Verykios, and Ahmed K. Elmagarmid. “TAILOR: A Record Linkage Toolbox”. In: *Proceedings of the 18th International Conference on Data Engineering*. IEEE. 2002, pp. 17–28.

[Esk00] Eleazar Eskin. “Anomaly Detection over Noisy Data using Learned Probability Distributions”. In: *Proceedings of ICML, 2000*. 2000.

[Est+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. “A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of KDD*. Vol. 96. 1996, pp. 226–231.

[Est02] Vladimir Estivill-Castro. “Why So Many Clustering Algorithms: A Position Paper”. In: *SIGKDD Explor. Newsl.* 4.1 (June 2002), pp. 65–75. ISSN: 1931-0145. DOI: 10.1145/568574.568575.

[Fak+10] Ali Fakeri-Tabrizi, Sabrina Tollari, Nicolas Usunier, and Patrick Gallinari. “Improving Image Annotation in Imbalanced Classification Problems with Ranking SVM”. In: *Multilingual Information Access Evaluation II. Multimedia Experiments*. Springer, 2010, pp. 291–294.

[FP99] Tom Fawcett and Foster Provost. “Activity Monitoring: Noticing Interesting Changes in Behavior”. In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge discovery and data mining*. ACM. 1999, pp. 53–62.

[Fay+96] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, eds. *Advances in Knowledge Discovery and Data Mining.* Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996. ISBN: 0-262-56097-6.

[Fir+10] Claudiu S. Firan, Mihai Georgescu, Wolfgang Nejdl, and Raluca Paiu. “Bringing Order to Your Photos: Event-driven Classification of Flickr Images Based on Social Knowledge”. In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management.* ACM. 2010, pp. 189–198.

[GY06] Mohamed Medhat Gaber and Philip S. Yu. “A Holistic Approach for Resource-Aware Adaptive Data Stream Mining”. In: *New Generation Computing* 25.1 (2006), pp. 95–115.

[GZK05] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. “Mining Data Streams: A Review”. In: *SIGMOD Rec.* 34.2 (June 2005), pp. 18–26. ISSN: 0163-5808. DOI: 10.1145/1083784.1083789.

[GH97] Robert Gaizauskas and Kevin Humphreys. “Using a Semantic Network for Information Extraction”. In: *Natural Language Engineering* 3 (02 Sept. 1997), pp. 147–169. ISSN: 1469-8110.

[Gan+02] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. “Sweetening Ontologies with DOLCE”. In: *Knowledge engineering and knowledge management: Ontologies and the semantic Web.* Springer, 2002, pp. 166–181.

[Gao+05] Jianfeng Gao, Haoliang Qi, Xinsong Xia, and Jian-Yun Nie. “Linear Discriminant Model for Information Retrieval”. In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM. 2005, pp. 290–297.

[Gao+10] Jing Gao, Feng Liang, Wei Fan, Chi Wang, Yizhou Sun, and Jiawei Han. “On Community Outliers and their Efficient Detection in Information Networks”. In: *KDD.* Ed. by Bharat Rao, Balaji Krishnapuram, Andrew Tomkins, and Yang Qiang. ACM, 2010, pp. 813–822. ISBN: 978-1-4503-0055-1.

[Gra+02] Adrian Graham, Hector Garcia-Molina, Andreas Paepcke, and Terry Winograd. “Time as Essence for Photo Browsing Through Personal Digital Libraries”. In: *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries.* ACM. 2002, pp. 326–335.

[Gru69] Frank E. Grubbs. “Procedures for Detecting Outlying Observations in Samples”. In: *Technometrics* 11.1 (1969), pp. 1–21.

[Guh+00] Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O’Callaghan. “Clustering Data Streams”. In: *Proceedings of the 41st Annual Symposium on Foundations of Computer Science, 2000.* IEEE. 2000, pp. 359–366.

[GRS98] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. “CURE: An Efficient Clustering Algorithm for Large Databases”. In: *SIGMOD Conference.* Ed. by Laura M. Haas and Ashutosh Tiwary. ACM Press, 1998, pp. 73–84. ISBN: 0-89791-995-5.

[GGC13] Itika Gupta, Kshitij Gautam, and Krishna Chandramouli. “VIT@MediaEval 2013 Social Event Detection Task: Semantic Structuring of Complementary Information for Clustering Events”. In: *MediaEval 2013 Workshop.* Spain, 2013.

[HBV01] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. “On Clustering Validation Techniques”. In: *Journal of Intelligent Information Systems* 17.2-3 (2001), pp. 107–145.

[HK06] Jiawei Han and Micheline Kamber. *Data Mining, Southeast Asia Edition: Concepts and Techniques.* Morgan Kaufmann, 2006.

[Har05] Stevan Harnad. “To Cognize is to Categorize: Cognition is Categorization”. In: *Handbook of Categorization in Cognitive Science.* Ed. by Henri Cohen and Claire Lefebvre. Elsevier, 2005.

[He+10] Bingsheng He, Mao Yang, Zhenyu Guo, Rishan Chen, Bing Su, Wei Lin, and Lidong Zhou. “Comet: Batched Stream Processing for Data Intensive Distributed Computing”. In: *Proceedings of the 1st ACM symposium on Cloud computing.* ACM. 2010, pp. 63–74.

[HCL07] Qi He, Kuiyu Chang, and Ee-Peng Lim. “Analyzing Feature Trajectories for Event Detection”. In: *SIGIR ’07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* Amsterdam, The Netherlands: ACM, 2007, pp. 207–214. ISBN: 978-1-59593-597-7.

[Hei71] Eleanor Heider. “”Focal” Color Areas and the Development of Color Names”. In: *Developmental Psychology* 4.3 (1971), p. 447.

[HA04] Victoria J. Hodge and Jim Austin. “A Survey of Outlier Detection Methodologies”. In: *Artificial Intelligence Review* 22.2 (2004), pp. 85–126.

[JMF99] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. “Data Clustering: A Review”. In: *ACM Comput. Surv.* 31.3 (Sept. 1999), pp. 264–323. ISSN: 0360-0300. DOI: 10.1145/331499.331504.

[JA07] Apirak Jirayusakul and Surapong Auwatanamongkol. “A Supervised Growing Neural Gas Algorithm for Cluster Analysis”. In: *International Journal of Hybrid Intelligent Systems* 4.2 (2007), pp. 129–141.

[Joa02] Thorsten Joachims. “Optimizing Search Engines using Clickthrough Data”. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM. 2002, pp. 133–142.

[Kim73] Jaegwon Kim. “Causation, Nomic Subsumption, and the Concept of Event”. In: *The Journal of Philosophy* (1973), pp. 217–236.

[Kim69] Jaegwon Kim. “Events and Their Descriptions: Some Considerations”. In: *Essays in Honor of Carl G. Hempel.* Ed. by Nicholas Rescher. Vol. 24. Synthese Library. Springer Netherlands, 1969, pp. 198–215. ISBN: 978-90-481-8332-6. DOI: 10.1007/978-94-017-1466-2_10.

[Kim76] Jaegwon Kim. *Events as Property Exemplifications.* Springer, 1976.

[Kim+09] Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. "Overview of BioNLP'09 Shared Task on Event Extraction". In: *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*. Association for Computational Linguistics. 2009, pp. 1–9.

[KR92] Kenji Kira and Larry A. Rendell. "A Practical Approach to Feature Selection". In: *Proceedings of the Ninth International Workshop on Machine Learning*. ML92. Aberdeen, Scotland, United Kingdom: Morgan Kaufmann Publishers Inc., 1992, pp. 249–256. ISBN: 1-5586-247-X.

[KKM02] Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. "From Instance-level Constraints to Space-level Constraints: Making the Most of Prior Knowledge in Data Clustering". In: (2002).

[KN98] Edwin M. Knorr and Raymond T. Ng. "Algorithms for Mining Distance-Based Outliers in Large Datasets". In: *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 392–403. ISBN: 1-55860-566-5.

[KK07] Igor Kononenko and Matjaž Kukar. *Machine Learning and Data Mining*. Elsevier, 2007.

[KSZ08] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. "Angle-Based Outlier Detection in High-dimensional Data". In: *KDD*. Ed. by Ying Li, Liu Bing, and Sunita Sarawagi. ACM, 2008, pp. 444–452. ISBN: 978-1-60558-193-4.

[KA04] Giridhar Kumaran and James Allan. "Text Classification and Named Entities for New Event Detection". In: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '04. Sheffield, United Kingdom: ACM, 2004, pp. 297–304. ISBN: 1-58113-881-4. DOI: 10.1145/1008992.1009044.

[Kun04] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, 2004.

[KZ08] Christopher A. Kurby and Jeffrey M. Zacks. "Segmentation in the Perception and Memory of Events". In: *Trends in Cognitive Sciences* 12.2 (2008), pp. 72–79. ISSN: 1364-6613. DOI: 10.1016/j.tics.2007.11.004.

[Lak82] George Lakoff. "Categories: An Essay in Cognitive Linguistics". In: *Linguistics in the Morning Calm* (1982), pp. 139–193.

[Lak87] George Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago and London: The University of Chicago Press, 1987.

[Lar+13] Martha A. Larson, Xavier Anguera, Timo Reuter, Gareth J. F. Jones, Bogdan Ionescu, Markus Schedl, Tomas Piatrik, Claudia Hauff, and Mohammad Soleymani, eds. *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop, Barcelona, Spain, October 18-19, 2013*. Vol. 1043. CEUR Workshop Proceedings. CEUR-WS.org, 2013. URL: http://ceur-ws.org/Vol-1043.

[LSR88] G. Daniel Lassiter, Julie I. Stone, and Scott L. Rogers. “Memorial Consequences of Variation in Behavior Perception”. In: *Journal of Experimental Social Psychology* 24.3 (1988), pp. 222–239.

[Lem67] John Lemmon. “Comments on The Logical Form of Action Sentences by Donald Davidson”. In: *The Logic of Decision and Action.* Ed. by Nicholas Rescher. Pittsburgh, 1967, pp. 96–103.

[Lew+06] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. “Content-based Multimedia Information Retrieval: State of the Art and Challenges”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)* 2.1 (2006), pp. 1–19.

[Lew97] David D. Lewis. *Reuters-21578, distribution 1.0 (Corpus).* 1997. URL: `http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`.

[Lew86] David K. Lewis. *On the Plurality of Worlds.* Vol. 322. Cambridge Univ Press, 1986.

[Lin+10] Cindy Xide Lin, Bo Zhao, Qiaozhu Mei, and Jiawei Han. “PET: A Statistical Model for Popular Events Tracking in Social Communities”. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD ’10. Washington, DC, USA: ACM, 2010, pp. 929–938. ISBN: 978-1-4503-0055-1. DOI: `10.1145/1835804.1835922`.

[LTH11] Xueliang Liu, Raphaël Troncy, and Benoit Huet. “Finding Media Illustrating Events”. In: *Proceedings of the 1st ACM International Conference on Multimedia Retrieval.* ICMR ’11. Trento, Italy: ACM, 2011, 58:1–58:8. ISBN: 978-1-4503-0336-1. DOI: `10.1145/1991996.1992054`.

[Lov11] John Lovett. *Social Media Metrics Secrets.* Vol. 159. John Wiley & Sons, 2011.

[Mac+67] James MacQueen et al. “Some Methods for Classification and Analysis of Multivariate Observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability.* Vol. 1. 14. California, USA. 1967, pp. 281–297.

[MHP11] Claudia Maienborn, Klaus von Heusinger, and Paul Portner. *Semantics: An International Handbook of Natural Language Meaning.* Vol. 33. Walter de Gruyter, 2011.

[Mak03] Juha Makkonen. “Investigations on Event Evolution in TDT”. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Proceedings of the HLT-NAACL 2003 Student Research Workshop-Volume 3.* Association for Computational Linguistics. 2003, pp. 43–48.

[MG13] Daniel Manchon-Vinzuete and Xavier Giro-i-Nieto. “UPC at MediaEval 2013 Social Event Detection Task”. In: *MediaEval 2013 Workshop.* Spain, 2013.

[MS03a] Markos Markou and Sameer Singh. “Novelty Detection: A Review—Part 1: Statistical Approaches”. In: *Signal processing* 83.12 (2003), pp. 2481–2497.

[MS03b] Markos Markou and Sameer Singh. “Novelty Detection: A Review—Part 2: Neural Network Based Approaches”. In: *Signal processing* 83.12 (2003), pp. 2499–2521.

[Mas+10] Mohammad M. Masud, Qing Chen, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani M. Thuraisingham. “Classification and Novel Class Detection of Data Streams in a Dynamic Feature Space.” In: *ECML/PKDD (2)*. Ed. by José L. Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag. Vol. 6322. Lecture Notes in Computer Science. Springer, 2010, pp. 337–352. ISBN: 978-3-642-15882-7.

[Mas+11] Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. “Classification and Novel Class Detection in Concept-Drifting Data Streams under Time Constraints”. In: *Knowledge and Data Engineering, IEEE Transactions on* 23.6 (2011), pp. 859–874.

[MNU00] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. “Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching”. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2000, pp. 169–178.

[McC93] Scott McCloud. *Understanding Comics: The Invisible Art*. Ed. by Mark Martin. Tundra Publishing, 1993.

[MK06] Matthew Michelson and Craig A. Knoblock. “Learning Blocking Schemes for Record Linkage”. In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 21. 1. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 2006, p. 440.

[Mit99] Tom M. Mitchell. “Machine Learning and Data Mining”. In: *Commun. ACM* 42.11 (Nov. 1999), pp. 30–36. ISSN: 0001-0782. DOI: 10.1145/319382.319388.

[Mur02] Gregory Leo Murphy. *The Big Book of Concepts*. MIT Press, 2002.

[Nal+08] Murilo Coelho Naldi, AndréC.P.L.F. de Carvalho, RicardoJoséGabrielliBarreto Campell, and eduardoRaul Hruschka. “Genetic Clustering for Data Mining”. In: *Soft Computing for Knowledge Discovery and Data Mining*. Ed. by Oded Maimon and Lior Rokach. Springer US, 2008, pp. 113–132. ISBN: 978-0-387-69934-9. DOI: 10.1007/978-0-387-69935-6_5.

[Nei86] Ulric Neisser. “Nested Structure in Autobiographical Memory”. In: *Autobiographical Memory*. Ed. by David Rubin. Wiley Online Library, 1986. Chap. 5.

[NE76] Darren Newtson and Gretchen Engquist. “The Perceptual Organization of Ongoing Behavior”. In: *Journal of Experimental Social Psychology* 12.5 (1976), pp. 436–450.

[NEB77] Darren Newtson, Gretchen A Engquist, and Joyce Bois. “The Objective Basis of Behavior Units”. In: *Journal of Personality and Social Psychology* 35.12 (1977), p. 847.

[New+87] Darren Newtson, Joan Hairfield, John Bloomingdale, and Steven Cutino. “The Structure of Action and Interaction”. In: *Social Cognition* 5.3 (1987), pp. 191–237.

[Ngu+13] Truc-Vien Nguyen, Minh-Son Dao, Riccardo Mattivi, Emanuele Sansone, Francesco GB De Natale, and Giulia Boato. “Event Clustering and Classification from Social Media: Watershed-based and Kernel Methods.” In: *MediaEval 2013 Workshop*. Spain, 2013.

[Ove+12] Paul Over, George Awad, Jonathan Fiscus, Greg Sanders, Barbara Shaw, Martial Michel, Alan F. Smeaton, Wessel Kraaij, Georges Quénot, et al. "TRECVID 2012 An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics". In: *Proceedings of TRECVID 2012*. 2012.

[Pap+13a] Symeon Papadopoulos, Emmanouil Schinas, Vasileios Mezaris, Raphaël Troncy, and Ioannis Kompatsiaris. "The 2012 Social Event Detection Dataset". In: *Proceedings of the 4th ACM Multimedia Systems Conference*. MMSys '13. Oslo, Norway: ACM, 2013, pp. 102–107. ISBN: 978-1-4503-1894-5. DOI: 10.1145/2483977.2483989.

[Pap+13b] Athanasios Papaoikonomou, Konstantinos Tserpes, Magdalini Kardara, and Theodora Varvarigou. "A similarity-based Chinese Restaurant Process for Social Event Detection". In: *MediaEval 2013 Workshop*. Spain, 2013.

[PA98] R. Papka and J. Allan. *On-Line New Event Detection using Single Pass Clustering*. Tech. rep. Amherst, MA, USA: University of Massachusetts, 1998.

[Pet+14] Georgios Petkos, Symeon Papadopoulos, Vasileios Mezaris, Raphael Troncy, Philipp Cimiano, Timo Reuter, and Yiannis Kompatsiaris. "Social Event Detection at MediaEval: A Three-year Retrospect of Tasks and Results". In: Proc. ACM ICMR 2014 Workshop on Social Events in Web Multimedia (SEWM). Glasgow, UK, April 2014, 2014.

[Pla99] John C. Platt. "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *Advances in Large Margin Classifiers*. 1999.

[PCF03] John C. Platt, Mary Czerwinski, and Brent A. Field. "PhotoTOC: Automatic Clustering for Browsing Personal Photographs". In: *Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia*. Vol. 1. IEEE. 2003, pp. 6–10.

[Qui85] Willard Van Orman Quine. "Events and Reification". In: *Actions and Events: Perspectives on the Philosophy of Donald Davidson* (1985), pp. 162–171.

[Raf+13] Dimitros Rafailidis, Theodoros Semertzidis, Michalis Lazaridis, Michael G. Strintzis, and Petros Daras. "A Data-Driven Approach for Social Event Detection". In: *MediaEval 2013 Workshop*. Spain, 2013.

[RRS00] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. "Efficient Algorithms for Mining Outliers from Large Data Sets". In: *SIGMOD Conference*. Ed. by Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein. SIGMOD Record 29(2), June 2000. ACM, 2000, pp. 427–438. ISBN: 1-58113-218-2.

[Rat+02] Gunnar Ratsch, Sebastian Mika, Bernhard Schölkopf, and K. Müller. "Constructing Boosting Algorithms from SVMs: An Application to One-class Classification". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.9 (2002), pp. 1184–1199.

[RGN07] Tye Rattenbury, Nathaniel Good, and Mor Naaman. "Towards Automatic Extraction of Event and Place Semantics from Flickr Tags". In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM. 2007, pp. 103–110.

[RN09] Tye Rattenbury and Mor Naaman. "Methods for Extracting Place Semantics from Flickr Tags". In: *ACM Transactions on the Web (TWEB)* 3.1 (2009).

[RS06] Steffen Rendle and Lars Schmidt-Thieme. "Object Identification with Constraints". In: *Sixth International Conference on Data Mining, 2006. ICDM'06.* IEEE. 2006, pp. 1026–1031.

[RS08] Steffen Rendle and Lars Schmidt-Thieme. "Scaling Record Linkage to Non-uniform Distributed Class Sizes". In: *Advances in Knowledge Discovery and Data Mining.* Springer, 2008, pp. 308–319.

[RC12] Timo Reuter and Philipp Cimiano. "Event-based Classification of Social Media Streams". In: Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR 2012). 2012.

[RC11] Timo Reuter and Philipp Cimiano. "Learning Similarity Functions for Event Identification using Support Vector Machines." In: *KDIR.* Ed. by Joaquim Filipe and Ana L. N. Fred. SciTePress, 2011, pp. 208–215. ISBN: 978-989-8425-79-9.

[Reu+14] Timo Reuter, Symeon Papadopoulos, Vasilios Mezaris, and Philipp Cimiano. "ReSEED: Social Event dEtection Dataset". In: *Proceedings of the 5th ACM Multimedia Systems Conference.* ACM. 2014, pp. 35–40.

[Reu+13] Timo Reuter, Symeon Papadopoulos, Giorgos Petkos, Vasileios Mezaris, Yiannis Kompatsiaris, Philipp Cimiano, Christopher de Vries, and Shlomo Geva. "Social Event Detection at MediaEval 2013: Challenges, Datasets, and Evaluation". In: *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop.* Proceedings of the MediaEval Multimedia Benchmark Workshop. Barcelona, Spain, 2013.

[Ric93] Michael M. Richter. "Classification and Learning of Similarity Measures". In: *Information and Classification.* Ed. by Otto Opitz, Berthold Lausen, and Rüdiger Klar. Studies in Classification, Data Analysis and Knowledge Organization. Springer Berlin Heidelberg, 1993, pp. 323–334. ISBN: 978-3-540-56736-3. DOI: `10.1007/978-3-642-50974-2_33`.

[Rob92] John A. Robinson. "First Experience Memories: Contexts and Functions in Personal Histories". In: *Theoretical Perspectives on Autobiographical Memory.* Springer, 1992, pp. 223–239.

[Ros75] Eleanor Rosch. "Cognitive Representations of Semantic Categories". In: *Journal of Experimental Psychology: General* 104.3 (1975), p. 192.

[RL78] Eleanor Rosch and Barbara B Lloyd. "Cognition and Categorization". In: *Hillsdale, New Jersey* (1978).

[RM75] Eleanor Rosch and Carolyn B. Mervis. "Family Resemblances: Studies in the Internal Structure of Categories". In: *Cognitive Psychology* 7.4 (1975), pp. 573–605. ISSN: 0010-0285. DOI: `10.1016/0010-0285(75)90024-9`.

[Sal91] Steven Salzberg. "A Nearest Hyperrectangle Learning Method". In: *Machine learning* 6.3 (1991), pp. 251–276.

[Sam+13] Sina Samangooei, Jonathan Hare, David Dupplaw, Mahesan Niranjan, Nicholas Gibbins, and Paul Lewis. "Social Event Detection via Sparse Multi-modal Feature Selection and Incremental Density based Clustering". In: *MediaEval 2013 Workshop*. Spain, 2013.

[San94] Mark Sanderson. "Reuters Test Collection". In: *BSC IRSG* (1994).

[Sar+04] Risto Sarvas, Mikko Viikari, Juha Pesonen, and Hanno Nevanlinna. "MobShare: Controlled and Immediate Sharing of Mobile Images". In: *Proceedings of the 12th Annual ACM International Conference on Multimedia*. ACM. 2004, pp. 724–731.

[Sav11] Eike von Savigny. *Ludwig Wittgenstein: Philosophische Untersuchungen*. Vol. 13. Walter de Gruyter, 2011.

[Sch+13] Emmanouil Schinas, Eleni Mantziou, Symeon Papadopoulos, Georgios Petkos, and Yiannis Kompatsiaris. "CERTH @ MediaEval 2013 Social Event Detection Task". In: *MediaEval 2013 Workshop*. Spain, 2013.

[SZ08] Thomas F. Shipley and Jeffrey M. Zacks. *Understanding Events: From Perception to Action*. Oxford Series in Visual Cognition. Oxford University Press, USA, 2008. ISBN: 9780198040705.

[SM81] Edward E. Smith and Douglas L. Medin. *Categories and Concepts*. Harvard University Press Cambridge, MA, 1981.

[SN01] Benno Stein and Oliver Niggemann. "Generation of Similarity Measures from Different Sources". In: *Engineering of Intelligent Systems*. Springer, 2001, pp. 197–206.

[Ste56] Hugo Steinhaus. "Sur la division des corps matériels en parties". In: *Bull. Acad. Polon. Sci. Cl. III. 4* (1956), pp. 801–804.

[SGM00] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. "Impact of Similarity Measures on Web-page Clustering". In: *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*. 2000, pp. 58–64.

[Sud76] Seymour Sudman. *Applied Sampling*. Academic Press New York, 1976.

[S+98] Cecilia Surace, K. Worden, et al. "A Novelty Detection Method to Diagnose Damage in Structures: An Application to an Offshore Platform". In: *The Eighth International Offshore and Polar Engineering Conference*. International Society of Offshore and Polar Engineers. 1998.

[SN13] Taufik Sutano and Richi Nayak. "ADMRG @ MediaEval 2013 Social Event Detection". In: *MediaEval 2013 Workshop*. Spain, 2013.

[Tat+03] Nesime Tatbul, Uğur Çetintemel, Stan Zdonik, Mitch Cherniack, and Michael Stonebraker. “Load Shedding in a Data Stream Manager”. In: *Proceedings of the 29th International Conference on Very Large Data Bases-Volume 29*. VLDB Endowment. 2003, pp. 309–320.

[TSS03] Dionyssios Theofilou, Volker Steuber, and Erik De Schutter. “Novelty Detection in a Kohonen-like Network with a Long-term Depression Learning Rule”. In: *Neurocomputing* 52 (2003), pp. 411–417.

[Tra14] TRECVID Multimedia Event Detection Track. *2014 TRECVID Multimedia Event Detection & Recounting Evaluation Plan*. 2014. URL: `http://nist.gov/itl/iad/mig/upload/MED_MER14_Evaluation_Plan_2014-05-15.pdf`.

[Tul72] Endel Tulving. “Episodic and Semantic Memory”. In: *Organization of Memory*. Ed. by Endel Tulving and Wayne Donaldson. Oxford: Academic Press, 1972, pp. 382–402.

[UF00] Tolga Urhan and Michael J. Franklin. “XJoin: A Reactively-Scheduled Pipelined Join Operator”. In: *Bulletin of the Technical Committee on* (2000), p. 27.

[Vaz01] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[VG04] Adam Vinueza and G. Grudic. *Unsupervised Outlier Detection and Semi-supervised Learning*. Tech. rep. Technical Report CU-CS-976-04, University of Colorado at Boulder, 2004.

[Vla+04] Michail Vlachos, Christopher Meek, Zografoula Vagena, and Dimitrios Gunopulos. “Identifying Similarities, Periodicities and Bursts for Online Search Queries”. In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. ACM. 2004, pp. 131–142.

[VGT12] Christopher M. De Vries, Shlomo Geva, and Andrew Trotman. “Document Clustering Evaluation: Divergence from a Random Baseline”. In: *CoRR* abs/1208.5654 (2012).

[Wag+01] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. “Constrained K-Means Clustering with Background Knowledge”. In: *ICML*. Vol. 1. 2001, pp. 577–584.

[WL11] Jianshu Weng and Bu-Sung Lee. “Event Detection in Twitter”. In: *ICWSM* 11 (2011), pp. 401–408.

[WCC07] Helen L. Williams, Martin A. Conway, and Gillian Cohen. “Autobiographical Memory”. In: *Memory in the Real World*. Ed. by Gillian Cohen and Martin A. Conway. 3rd ed. East Sussex/New York: Psychology Press, 2007. Chap. 2, pp. 21–90. ISBN: 978-1-84169-641-9.

[WS13] Martin Wistuba and Lars Schmidt-Thieme. “Supervised Clustering of Social Media Streams”. In: *MediaEval 2013 Workshop*. Spain, 2013.

[Xin+02] Eric P. Xing, Michael I. Jordan, Stuart Russell, and Andrew Y. Ng. “Distance Metric Learning with Application to Clustering with Side-information”. In: *Advances in Neural Information Processing Systems*. 2002, pp. 505–512.

[X+05] Rui Xu, Donald Wunsch, et al. "Survey of Clustering Algorithms". In: *Neural Networks, IEEE Transactions on* 16.3 (2005), pp. 645–678.

[YPC98] Yiming Yang, Tom Pierce, and Jaime Carbonell. "A Study of Retrospective and On-line Event Detection". In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM. 1998, pp. 28–36.

[YC01] Nong Ye and Qiang Chen. "An Anomaly Detection Technique Based on a Chi-Square Statistic for Detecting Intrusions into Information Systems". In: *Quality and Reliability Engineering International* 17.2 (2001), pp. 105–112.

[ZT01] Jeffrey M. Zacks and Barbara Tversky. "Event Structure in Perception and Conception". In: *Psychological Bulletin* 127.1 (2001), p. 3.

[ZTI01] Jeffrey M. Zacks, Barbara Tversky, and Gowri Iyer. "Perceiving, Remembering, and Communicating Structure in Events". In: *Journal of Experimental Psychology: General* 130.1 (2001).

[ZZD13] Matthias Zeppelzauer, Maia Zaharieva, and Manfred Del Fabro. "Unsupervised Clustering of Social Events". In: *MediaEval 2013 Workshop.* Spain, 2013.

[ZZW07] Kuo Zhang, Juan Zi, and Li Gang Wu. "New Event Detection Based on Indexing-tree and Named Entity". In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '07. Amsterdam, The Netherlands: ACM, 2007, pp. 215–222. ISBN: 978-1-59593-597-7. DOI: 10.1145/1277741.1277780.

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. "BIRCH: An Efficient Data Clustering Method for Very Large Databases". In: *ACM SIGMOD Record.* Vol. 25. 2. ACM. 1996, pp. 103–114.