German Research Center for Artificial Intelligence

Document D-14-01

Readings on Artificial Intelligence Applications

Proceedings of the Seminar “Collaborative Intelligence” 2013/2014

Andreas Dengel, Darko Obradovic, 04/2014

Document D-14-01
German Research Center for Artificial Intelligence (DFKI) GmbH

Bibliographic information published by the German National Library

The German National Library lists this publication in the German National Biography; detailed bibliographic data are available in the internet at http://dnb.ddb.de

Issue D-14-01 (2014)
ISSN 0946-0098

German Research Center for Artificial Intelligence
Deutsches Forschungszentrum für Künstliche Intelligenz
DFKI GmbH

Founded in 1988, DFKI today is one of the largest nonprofit contract research institutes in the field of innovative software technology based on Artificial Intelligence (AI) methods. DFKI is focusing on the complete cycle of innovation – from world-class basic research and technology development through leading-edge demonstrators and prototypes to product functions and commercialization.

Based in Kaiserslautern, Saarbrücken and Bremen, the German Research Center for Artificial Intelligence ranks among the important 'Centers of Excellence' worldwide. An important element of DFKI's mission is to move innovations as quickly as possible from the lab into the marketplace. Only by maintaining research projects at the forefront of science DFKI has the strength to meet its technology transfer goals.

The key directors of DFKI are Prof. Wolfgang Wahlster (CEO) and Dr. Walter Olthoff (CFO). DFKI's research departments are directed by internationally recognized research scientists:

- Knowledge Management (Prof. A. Dengel)
- Cyber-Physical Systems (Prof. R. Drechsler)
- Robotics Innovation Center (Prof. F. Kirchner)
- Innovative Retail Laboratory (Prof. A. Krüger)
- Institute for Information Systems (Prof. P. Loos)
- Embedded Intelligence (Prof. P. Lukowicz)
- Agents and Simulated Reality (Prof. P. Slusallek)
- Augmented Vision (Prof. D. Stricker)
- Language Technology (Prof. H. Uszkoreit)
- Intelligent User Interfaces (Prof. W. Wahlster)
- Innovative Factory Systems (Prof. D. Zühlke)

In this series, DFKI publishes research reports, technical memos, documents (eg. workshop proceedings), and final project reports. The aim is to make new results, ideas, and software available as quickly as possible.

Prof. Wolfgang Wahlster
Director

Readings on Artificial Intelligence Applications

Proceedings of the Seminar "Collaborative Intelligence" 2013/2014

Andreas Dengel, Darko Obradovic

04/2014

Document D-14-01 des
Deutschen Forschungszentrums für Künstliche Intelligenz (DFKI)

Abstract

This document contains the articles written by our students in the Seminar "Collaborative Intelligence" 2013/2014. The Seminar was organized as a scientific mini-conference with peer-reviewed submissions that present state-of-the-art approaches in Collaborative Intelligence technologies. The four accepted articles cover different topics: event detection in Social Media, keyword-based semantic search, feature-based object detection and recognition, and – last but not least – semantic technologies for personal memory support.

Zusammenfassung

Dieses Dokument enthält die Artikel unserer Studierenden aus dem Seminar "Collaborative Intelligence" 2013/2014. Das Seminar wurde als wissenschaftliche Mini-Konferenz mit begutachteten Einreichungen zu aktuellen Ansätzen aus dem Bereich "Collaborative Intelligence" durchgeführt. Die vier akzeptierten Artikel umfassen ein breites Themenspektrum: die Detektion von Ereignissen in Sozialen Medien, schlüsselwort-basierte semantische Suche, die merkmal-basierte Erkennung von Objekten, sowie die Unterstützung persönlicher Gedächtnisse mit Hilfe semantischer Technologien.

Preface

Readings on Artificial Intelligence Applications is a collection of papers that are written for the Seminars of the Knowledge Management (KM) group being part of the Computer Science program at the University of Kaiserslautern. The Seminar is held at DFKI in Kaiserslautern.

Our intention is to provide a forum in which students may be introduced into scientific work as it is a matter of fact when publishing research papers at international conferences or workshops. Consequently, students have to investigate defined topics and write papers following given guidelines. We install a program committee consisting of the supervisor team and the students participating in the Seminar. The individual contributions (submissions) have been peer-reviewed using the criteria that are common in international research communities. This reviewing process not only increases the quality of the contributions by giving rich feedback to the authors but also makes the seminar similar to a workshop broadening the experience of the students.

This semester we have accepted four articles that are presented in a workshop-like session. They are all well written and describe state-of-the-art approaches in the area of Artificial Intelligence. We hope that other researches may profit from this collection and like to thank all authors for their collaboration and their excellent contributions.

April 2014 Prof. Dr. Andreas Dengel, Dr. Darko Obradovic

Reviewers

Sheraz Ahmed
Ali Bahrainian
Björn Forcher
Heiko Maus
Sepideh Saran
Rotem Stram
Matthias Streuber
Kristin Suchner

Table of Contents

Public Event Detection Using Online Social Media 1
Sepideh Saran

Keyword-Based Semantic Search 11
Rotem Stram

Part-based Features for Object Detection and Recognition 23
Matthias Streuber

Realizing Personal Memories with Semantic Technologies 35
Kristin Suchner

Public Event Detection Using Online Social Media

Sepideh Saran
saran.sepideh@gmail.com

Computer Science Department, University of Technology Kaiserslautern, Germany

Abstract. Social microblogging websites such as Twitter provide a platform for sharing daily observations and thoughts among millions of users. They are important data sources which contain detailed and real-time information. This data can be used to discover events occurring in daily life. This paper reviews some of the most important previous research work on this topic and explains a selected technique named Semi-Supervised Targeted Event Detection (STED). STED automatically analyzes tweets for detecting and describing public events on Twitter in real-time. This method, unlike many existing data mining algorithms, does not need extensive manual data labeling.

Keywords: Data Mining , Event Detection , Online Social Media Analysis, Twitter Analysis

1 Introduction

The rise of online social media has leaded to emergence of many new research fields. One such research field is event detection using the online social media. Events are real-world occurrences that can be described by a topic, time, location and scale. Researchers have focused on event detection using Twitter, as well as, other social networking websites. On Twitter, events could be defined as topics which become popular in some specific geographical regions in a certain time period. Tweets can contain detailed information about any topic. However, for extracting that information, traditional text retrieval techniques do not lead to acceptable results. That is because the language used in tweets, unlike traditional media, is informal, abbreviated and with lots of spelling and grammatical errors. Tweets are restricted in length, not well structured and may contain useless content.

In this paper we are focusing on detecting unknown new events as they occur, so we are eliminating already happened events. Events that have occurred in past, can be retrieved by traditional methods such as filtering and query generation. Twitter's active users report and share information instantly. It can be assumed that the time of an event is the same as the time when a considerable number of tweets about the event's topic are posted. In general we can divide the event detection process among tweets of a specific time period to three steps. The first step would be to identify tweets which are related to a real-world event

topic, the second step is to identify the location of the event and the last step is to define the scale and importance of the event (e.g. street-level, city-level).

Different approaches are proposed for each level, but in most previous work location detection is considered while clustering tweets to different topics. Few studies specifically paid attention to event scale as a describing factor. Different events might have different popularity among users and can differ in content, number of tweets, participants, period, inherent structure or casual relationship[5].

In the rest of paper, in Section 2 some of the most significant methods presented in previous work are introduced. Section 3 provides explanation of a selected event detection approach. Finally in Section 4 a general conclusion of this study is presented.

2 Related Work

Farzindar and Wael [5] reviewed some Twitter specified event detection techniques in a survey. They classified these techniques, depending on the type of events to unspecified and specified. According to the detection task and target application, methods are classified to New Event Detection (NED) and Retrospective Event Detection (RED). Furthermore, depending on the detection method, they are categorized into supervised and unsupervised methods. They suggest that unsupervised methods are more likely to have a better result for unspecified event detection.

In this study we are focusing on finding events that are new, unknown and unspecified. There is no information provided about unspecified events before finding them. Here we first briefly introduce a number of researches on finding known or partially known events. Afterwards we discuss some methods of detecting unknown events in social media.

2.1 Detecting Known or Partially Known Events

Becker et al. [3] focused on automatically identifying user-contributed content for planned event, therefore known in advance, across different social media websites. In their method, first queries for known event search are created. Secondly precision-oriented query are generated using known event features. Finally a recall-oriented query is developed to improve low recall of the precision-oriented strategies. They show this query which is generated based on a specific social media can be used to retrieve information about the event in other social media websites. Tweets are short in length and some aspects of an event such as location, can not be entirely retrieved clearly from tweets. Hence, this work can be useful to collect more information describing the same event from other social media sites along with Twitter.

Popescu and Pennachhiotti [14] proposed a supervised approach using manually labeled data set for detecting controversial events about which users express opposing ideas (which are not considered as real-world events in our definition),

by assigning controversy scores to tweet snapshots and ranking them. Another study, focused on detecting specific types of events such as earthquake detection, proposed by Sakaki et al. They used manually labeled data to train a Support Vector Machine (SVM) to classify tweets expressing events from non-event tweets, using three types of features, namely, number of words, keywords and contextual words in a query [5]. But the problem with most supervised methods is that they are not applicable for detecting unknown events, because they need large amount of manually labeled data.

2.2 Detecting Unknown Events

Several methods are presented to detect events which we do not have any previous knowledge about. These methods either used real-time clustering or offline processing for identifying tweets which report an event from non-event tweets.

Online Clustering Approaches :
Sankaranarayanan et al. [15], proposed a system called TwitterStrand, to automatically obtain breaking news from tweets. They also provided a map interface for visual representation of the found event. Tweet's geo-tag locations and the location mentions within the tweets, are important factors in clustering tweets to events. They measure importance of an event by the number of posted tweets on a related topic and the speed of finding tweets in a cluster. According to a research done in 2009, less than 10% of users are responsible for 90% of tweets[7]. Based on this assumption, TwitterStrand finds a group of users called seekers who mainly report news. In the next step, it looks for most common set of followers among seekers. 2,000 users are identified manually to be news reporters, such as newspapers, television and bloggers. New users who seem to be interested in news tweets are added to seekers group and inactive users are removed constantly. TwitterStrand also uses links to external sources mentioned in seeders feed and it also enables search in news tweets by keywords. They use naive bayes classifier that is trained on manually labeled data to cluster tweets to news and non-news tweets then they apply an online clustering algorithm using a weighted feature vector which is weighted using the TF-IDF (Term FrequencyInverse Document Frequency) measure and cosine similarity to cluster news-tweets to different topics. Topic hashtags are used to reduce errors.

Phuvipadawat and Murata [13] proposed a method to collect, group, rank and track breaking news in Twitter and developed an application called Hotstream based on that method. For analyzing characteristics of breaking news, they collected a 121,000 public tweets and 33,000 tweets from users who use breaking news hashtag (i.e. #breakingnews) in their message. In their definition each tweet can have two important elements: facts and emoticons. Emoticons are not considered in this work.Their methodology is to first fetch tweets through the Twitter streaming API using predefined search queries with keywords like #breakingnews, then to index messages using Apache Lucene[1] which is a full-

[1] http://lucene.apache.org

featured text search engine library and then to group the messages using TF-IDF similarity measure. After forming clusters, representing events, each topic will be adjusted with a ranking system based on number of followers and number of retweets. New messages will join a cluster if they are similar to the first top-k items in the cluster. As many tweets about ongoing events do not include the term "breaking news". this method withdraws many useful tweets.

Petrovic et al. [12] presented an online method based on Locality-Sensitive Hashing (LSH) and experimented their method on 160 million tweets. This method relies on hashing each query point into buckets in such way that the probability of collision is much higher for points that are close to each other. Cosine similarity is used as a score for finding nearest tweets. There are a finite number of buckets and the number of documents within each bucket is limited to a constant. New document will replace the oldest one, if the bucket is full. After clustering tweets to threads, they consider fastest growing threads to represent an event. The growth rate shows the scale of the event. This work shows the number of users is a better measure for ranking than the number of tweets.

Becker et al. [2] used an online-clustering technique to cluster tweets to groups each representing a topic and then removed the non-event topics by applying an SVM classifier. The SVM was trained on manually labeled set of cluster features. Clustering approach is based on threshold-based incremental clustering. Threshold parameter is tuned during training phase. Each tweet is a weighted term vector using the TF-IDF measure and cosine similarity is used to form clusters. Stop-words are removed and hashtag words are doubled in weight. They assumed that event clusters are revolving around some few keywords while non-event clusters center around different terms. Cluster features which are used in identifying event clusters are Temporal (e.g. volume of frequent cluster terms), Social (e.g. retweet, reply and mention), Topical (e.g. common terms repeated frequently in cluster) and Twitter-centric features (e.g. tag and multiwordhashtags).

Long et al. [11] proposed a unified workflow of event detection, tracking and summarization on Sina[2], a popular microblogging approach in china. In this method, firstly, topical words are extracted based on word frequency, hashtags and word entropy. In the next step, related topical words are grouped to form a cluster representing an event by applying a hierarchical divisive clustering approach on a co-occurrence graph to connect words which co-occurred frequently and form k clusters of events. Bipartite graph is used for event tracking task. Related events are chained using the maximum-weight bipartite graph matching algorithm. Most related posts are selected to summarize the event.

Walther and Kaisser (2013) [16] proposed a method for real-world event detection in a monitored geographic area. They used their method mainly for detecting small-scale and local events. First, most recent tweets are stored into a MongoDB table. In the next step, if the number of tweets issued in specific time interval in a location of a constant radius is more than some constant, clusters are created. New tweets are constantly added to clusters; overlapping clusters are

[2] http://t.sina.com.cn

merged and clusters with old tweets are deleted. A supervised machine learning approach is used to identify event clusters using manually labeled training data. Some textual features (common words, near duplicates, positive and negative sentiment, hashtags, mentions and retweets) and non-textual features (tweet count, user count, link Ratio and Unique locations) are used to score tweets to pick the ones with higher score for summarizing the event.

Abdelhaq et al. [1] present a framework to detect localized events in real-time by adopting a continuous analysis of the most recent tweets within time-based sliding window and track their evolution over time. Events are described by number of related keywords, start time and location. All collected tweets are used to identify keywords using discrepancy paradigm based on their burstiness, but only geo-tagged tweets are used to estimate location. A single-pass clustering algorithm is used to cluster keywords to groups identifying events [1]. A score is given to each cluster to detect event clusters from non-event ones. Keywords of an event cluster are assumed to have high burstiness degree and are member of the cluster for a long time.

Offline processing Approaches :
Weng et al. [17] introduced EDCoW (Event Detection with Clustering Wavelet-based Signals). First, discrete signals for individual words are built which capture only the bursts in the word's appearance. Signal construction is based on DF-IDF (Document Frequency- Inverse Document Frequency) and a sliding window is used for capturing changes over time. DF is the counterpart of TF in TF-IDF. The signals can be fast computed by wavelet analysis and require less space for storage. In the next step, cross correlation between signals is measured and trivial words are removed. Signals are clustered to groups representing events, using a modularity-based graph partitioning. Finally, scale of the event is retrieved using number of words and cross correlation between them within each cluster.

Cordeiro [4] presented an event detection method on Twitter using continuous wavelet transformation. In this method, signals are created only based on hashtag words. First step is to retrieve hashtags mentioned in tweets and then group them in intervals of five minutes. Hashtags within every five intervals are counted and they are grouped in separate time series for each hashtag. Tweets with common hashtags are connected to each other within each of time series. The map reduce transformation, produces one signal for each hashtag in one time interval. The continuous wavelet transform (CWT) constructs a time-frequency representation of signals. Peak analysis is used to find peaks in hashtag signals and local maxima detection is used for detecting changes. Finally LDA (Latent Dirichlet Allocation) algorithm is used to create a summarization for describing the event.

Lee and Sumiya presented a local event detection method based on geo-tags[5]. Tweets with geo-tags are about two percentage of all the tweets. So the method can not provide good result and it is expected to have a low recall.

Li et al. [9] presented a Twitter based Event Detection and Analysis System (TEDAS) to detect Crime and Disaster related Events (CDE) in Twitter.

They analyze the spatial and temporal pattern of the event and identify the importance of the event. TEDAS contains two major parts, offline processing and online computing. Offline processing retrieves, processes and stores crime and disaster related tweets. Online processing answers user queries and generates visual results. CDE-crawlers retrieve related tweets using a set of well-defined keywords and some primary rules and new rules are also added based on retrieved tweets. A rule validator is also used to examine usefulness of rules based on ratio of new CDE tweets to new non-CDE tweets. To identify CDE tweets, a classifier is used based on five types of features. First, Twitter-specific features such as links and mentions. Second, CDE-specific features for example containing time or location. Third, Content features such as containing important words. Forth User features, for example whether a user is an authority or not. Finally, Usage features such as retweets. Tweets are ranked according to their importance by a learning-to-rank approach using a linear regression model.

Liu et al. [10] through analysis of retweets showed that changes in information flows are related to a real-time event [10]. They focused on geographical viewpoints of retweets and chose time-zone as indicator for location of users. In this method, bidirectional retweets between two different time zones are collected in specific dates and tag clouds are generated. Strongest flows between two locations are considered as news.

3 Event Detection Using An Online Clustering Method

As we already mentioned we divide event detection process to three stages. First, Topic detection is to find out that users are tweeting about some ongoing event. Second, Location estimation, and third, scale estimation to show how big or important the event is. In this section an event detection method called Semi-Supervised Targeted Event Detection (STED) presented by Hua et al.[6] is explained and evaluated. As shown in Figure 1, this model first applies transfer learning and label propagation to automatically generate labeled data. In the next step, learns a customized text classifier based on mini-clustering, and finally applies fast spatial scan statistics to estimate the location of the event. STED takes a topic of interest as an input and retrieves related events and summarizes event description.

3.1 Topic detection

STED contains an automatic labeling system which transfers labels from newspapers to tweets. Labels are expanded using social features of Twitter such as retweets, hashtag words and mentions. In the next stage, using tweet miniclusters obtained by graph partitioning, a specialized SVM classifier is built.

Automatic Labeling In this stage, domain specific news descriptions are collected. Afterwards, named entity (i.e. nouns) and action words (i.e. verbs) that

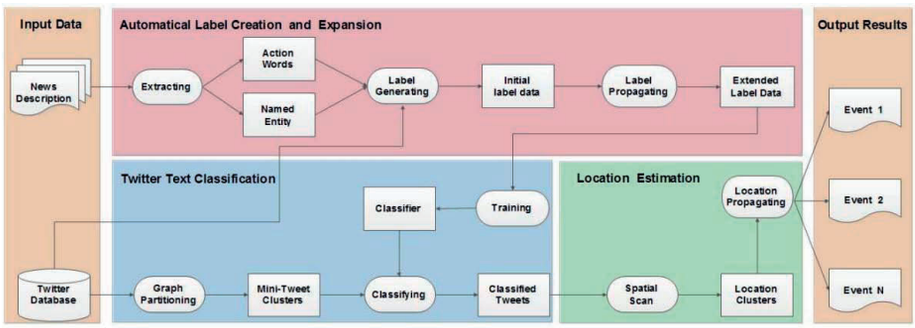
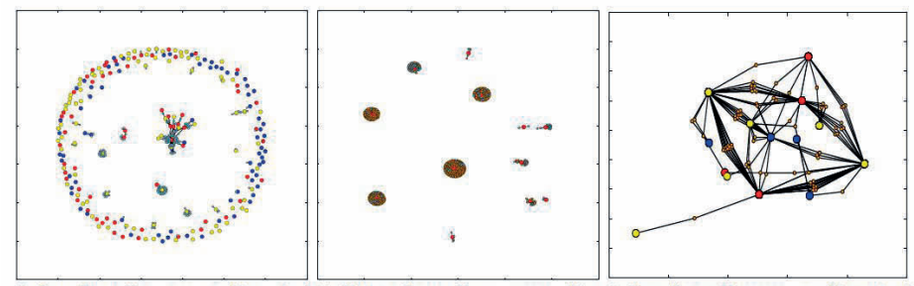


Fig. 1. System Framework of STED [6]

mainly describe events are extracted from news. For this task, NLTK[3]is used. NLTK is a Natural Language Processing Toolkit in Python. Tweets which contain at least one Named Entity or Action Word are labeled as event-related tweets. To expand labels, social-tie terms such as mentions, retweets and hashtag are used. Social ties in labeled tweets are identified, an as shown in Figure 2, a term-tweet heterogeneous network named S1 is built. In S1, Where most tweets are connected to few terms it is expected that those tweets are reporting an event. Less popular terms are removed and the remaining terms are used to generate queries. In the next step, a hashtag-tweet heterogeneous network named S2 is built to identify and then remove words which are popular but are common between too many topics. Finally a S3 term-tweet network with filtered terms connected to the newly found label tweets is built. This process iterates till no new tweet can be found and during label propagation, an extended label dataset is obtained and used for further processing.

Text Classification In this part, first graph partitioning methods are used for identifying event-related words. In the next step, an SVM is used for classification. For each word, its wavelet signal is built using TF-IDF score of the word in time intervals of one hour. Words that are repeating similarly every day are removed according to auto correlation of all words. Remaining words are considered to be noteworthy words. A correlation matrix is constructed based on cross-correlation of each word-pair. This matrix can be viewed as a graph and using graph partitioning we obtain subgraphs each including similar words. In generating clusters, tweets containing at least two words in the same subgraph will be items of the same cluster. In training phase of the classifier, words appearing less than specific threshold are removed. Finally, TF-IDF scores of words are calculated and frequent words are removed.

[3] www.nltk.org



(a) Term-Tweet Heterogeneous Network of Label Tweets S_1.

(b) Hashtag-Tweet Heterogeneous Network of Total Tweets Space S_2.

(c) Term-Tweet Heterogeneous Network of New Found Tweets and Filtered Terms S_3.

Fig. 2. Tweet's Social Ties Networks : Big nodes represent terms: Red nodes are hashtags, blue nodes are mentions and yellow nodes are retweets. Small nodes are tweets: blue ones are labeled tweets, orange nodes are newly found tweets from raw data. [6]

3.2 Location Estimation

Geo-locations of tweets about a certain event are likely to be close to location of the happening event. STED first applies spatial scan statistics to identify significant spatial clusters and then further propagates geo-labels within each cluster. Number of event-related tweets in each city is counted and number of original tweets in the city is defined as the base of the city. A fast subset scan is applied to identify clusters with largest kulldorfs statistic[8]which is defined as Equation 1[6].

$$K_r = (C_a - C_r)\, lg\left(\frac{C_a - C_r}{B_a - B_r}\right) + C_r lg\left(\frac{C_r}{B_r}\right) - C_a lg\left(\frac{C_a}{B_a}\right) \tag{1}$$

C_a and B_a refer to total count and base in a country; C_r and B_r refer to the count and base in the spatial region. Clusters with empirical p-value estimated by random permutation testing, that are smaller than threshold (e.g.0.05) are considered significant. Insignificant clusters are removed by randomization testing. As tweets containing common terms and hashtags within each cluster are more likely to have the same location, social ties are used to estimate location for big set of non-labeled tweets based on tweets with geo-tags. First a score is computed for tweets with geo-tags to show relativity of terms to locations, using the ratio of number of tweets containing a term in a specific location to total number of tweets containing that term. Then locations of unlabeled tweets are estimated by picking the biggest value for relativity between tweet and all locations in a dataset.

This method is applied to tweets from Latin America with a database size of 400GB and achieved 72% in precision and 74% in recall, with a lead time of 2.42 days ahead of traditional media. STED's user interface programmed in Python, shows map visualization of events, and presents an event summary based on most related words.

3.3 scale estimation

STED does not present a specific scale estimation method. Factors such as number of tweets in event cluster, number active users in each cluster, number of retweets, number followers of most active users and number of multiword hashtags and size of estimated location can be used to determine the importance of the event as a describing factor.

4 Conclusion

Event detection in online social media is the process of finding and describing real-world occurrences through processing user's posted messages in popular microblogging websites such as Twitter. Tweets are posted and received in real-time and include important user generated information, thus can be a useful source for detecting new unknown events. As tweets are brief and might have grammatical errors, traditional text processing approaches are not likely to present good results. Several supervised and unsupervised methods are presented to group tweets reporting about same topics and to identify event-related topics and find event description factors such as location and scale. As there is no information provided about unspecified events, unsupervised detection methods and hybrid approaches seem to be more efficient than supervised methods, because supervised approaches need large datasets of manually labeled data. In different levels of clustering, methods which first considered all words in a tweet and then marked hashtag words as more important terms and then removed less related tweets are more efficient than methods which only consider hashtag words and ignore the rest or the ones who do not pay special attention to hashtags at all. Twitter specific social ties such as retweets and mentions and user specific features such as number of followers can also be useful in both topic detection and scale estimation. For future work, it is suggested to design a hierarchical location estimation method which uses all three factors of geo-tags and location keywords such as city names and user's time zone for identifying location of the event. As each microblogging website has its own limitations, it is also recommended to track posted massages from different online social media websites simultaneously. This way we can receive more accurate and complete data about different aspects of an event. An automatically generated list of event-related keywords can also be useful to mark some words in tweets as important and weight them for further processing.

References

1. Abdelhaq H., Sengstock C., Gertz M.: EvenTweet: Online Localized Event Detection from Twitter. In the 39th International Conference on Very Larg Data Bases, Proceedings of the VLDB Endowment, Vol6, No. 12 (2013)
2. Becker H., Iter D., Naaman M., Gravano L.: Beyond trending Topics: Real-Word Event Identification on Twitter. In ICWSM, Barcelona, Spain (2011)

3. Becker H., Iter D., Naaman M., Gravano L.: Identifying Content for Planned Events Across Social Media Sites. In WSDM , Seattle, Washington, USA (2012)
4. Cordeiro M.: Twitter event detection: combining wavelet analysis and topic inference summarization. In proceedings of the 7th Doctoral Symposium In Informatics Engineering, pages 123-138 (2012)
5. Farzindar, A.,Wael K.: A Survey of Techniques for Event Detection in Twitter. Wiley Periodicals, Computational Intelligence journal (2013)
6. Hau T., Chen F., Zhao L., Lu C., Ramakrishnan N.: STED: Semi-Supervised Targeted Event Detection. KDD, Chicago, USA (2013)
7. Heil B., Piskorski M.: New Twitter Research: Men Follow Men and Nobody Tweets: Harvard Business Review, http://blogs.hbr.org/2009/06/new-twitter-research-men-follo/ (2009)
8. Kulldorf M.: Spatial scan statistic: models, calculations, and applications. Pages : 303-322. Springer 1999
9. Li R., Lei K., Khadiwala R., Cheng K.: TEDAS: a Twitter Based Event Detection and Analysis System.Data engineering (ICDE) , IEEE 28th International Conference (2012)
10. Liu P., Tang J., Wang T.: Information Current in Twitter: Which Brings Hot Events to the World. WWW Companion, Rio do Janeiro, Brazil, ACM 978-1-4503-2038-2/13/05 (2013)
11. Long R., Wag H., Chen Y., Jin O., Yu Y.: Towards Effective Event Detection, Tracking and Summarization on Microblog Data. In Web-age Information Management, Vol. 6897 of Lecture Notes in Computer Science, Springer: Berlin/Heidelberg (2011)
12. Petrovic S., Osborne M., Lavrenko V.: Streaming First Story Detection with application to Twitter. In Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (2010)
13. Phuvipadawat S., Murata T.: Breaking News Detection and Tracking in Twitter. In IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Vol. 3, Toronto, Canada (2010)
14. Popescu A., Pennacchiotti M.: Detecting Controversial Events from Twitter. In proceedings of the 19th ACM International Conference on the Information and Knowledge Management, CIKM10 , ACM, New York, USA (2010)
15. Sankaranarayanan J., Samet H., Teitler B., Lieberman M., Sperling J.: TwiiterStrand: News in Tweets. In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS09 , ACM, New York, USA (2009)
16. Walther M., Kaisser M.: Geo-spatial Event Detection in the Twitter Stream. ECIR, LNCS 7814, pp 356-357 , Springer (2013)
17. Weng J., Yao Y., Leonardi E., Lee B.: Event Detection in Twitter. In ICWSM , Barcelona, Spain (2011)

Keyword-Based Semantic Search

Rotem Stram
rotemstram@yahoo.com

Computer Science Department, University of Technology Kaiserslautern, Germany

Abstract. In an information retrieval (IR) setting, a user would like to find documents that are described by a few keywords he has given the system. The advantage of search by keywords is that the user does not need to understand the data structure or the query language (e.g. SQL). In addition, where traditional models allow only for syntactical retrieval, datasets in the form of semantic graphs have expanded the search possibilities. In order to accommodate this, new algorithms are needed to help retrieval systems map keywords to known elements in the graph. Next, finding the subgraph that connects those entities is necessary. This ensures these elements are related to each other and helps measure the quality of this relation. In this paper, two such algorithms will be presented. The first method uses graph traversal in order to find the needed connections. The second method divides the graph into neighbourhoods and performs graph joins on them in order to connect them and find a path between all keyword elements.

1 Introduction

In the last twenty years information retrieval and keyword searches have become an integral part of our lives. With the rise of web search giants such as Google[1] we can no longer imagine our lives without entering a query into our favourite engine whenever we are faced with a question we cannot answer. Traditional IR systems use the keywords to look for their presence in a document with the help of methods such as TF-IDF and the vector model [8] [7], but lately there has been a shift towards semantic technologies in order to identify the semantic meaning of the keywords. This means that instead of looking for an appearance of a word in a document the focus here is on identifying entities on a semantic graph.

The process of keyword search on the semantic graphs begins with interpreting the keywords according to the user's intent. This is done by finding a list of matching candidate graph elements in the index for each keyword. The process continues with a corresponding query generation by finding subgraphs which connect the found graph element and transforming them into a query in a language suitable for retrieval, such as SPARQL. Several approaches have been used for keyword search over graph structured data [3] [4] [1]. This paper will present two approaches for finding subgraphs by mapping keywords to data

[1] https://www.google.com/

elements, finding a connecting path between them, and outputting the top-k substructures according to some scoring function.

Two articles will be discussed in this paper, namely *Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data* [10] and *Index Structures and Top-k Join Algorithms for Native Keyword Search Databases* [5]. In the first article, the authors described an algorithm to calculate the top-k queries the user might have meant, as opposed to calculating the results. Their algorithm takes unstructured keywords as an input, finds the corresponding data elements candidates and then proceeds to find the top-k substructures connecting all keywords using graph traversal. Structured queries are then constructed from the resulting subgraphs. In the second paper, on the other hand, a neighbourhood for each node in the graph is constructed and graph joins are used in order to find the connecting subgraphs.

This paper is constructed as follows: section 2 will describe the algorithm presented in [10] along with an evaluation of it. Section 3 contains the description of the method presented in [5] and its evaluation. Section 4 will then conclude this paper.

2 Search for Top-k Query Candidates

In this section, the algorithm suggested by Tran et. al. [10] will be described. First a short overview of the algorithm will be given, then a description of each step, and at the end an evaluation will follow.

2.1 Overview

The novelty of the article [10] is in finding the top-k queries from a semantic graph without the need to calculate all possible results, and ordering them by their score. To this end, a new way of traversing the semantic graph is presented. The algorithm contains several steps. First, keywords inserted by the user are mapped to graph elements stored in the index. This results in a list of possible keyword elements, which are then used to find other elements that connects them (the so called connecting elements). The connecting elements are then part of a path that connects the keywords, and with them a matching subgraph is constructed. For each such subgraph a matching query is constructed through the derivation of the matching graph elements to query elements. This process is repeated until the top-k queries are computed.

2.2 Indexing Process

Indexing of the graph data is divided into two parts. First, an inverted index is created which maps labels of elements to their corresponding graph entities (both vertices and edges). Lexical analysis such as stemming and stop word elimination is performed, semantic similarities such as synonyms and hyponyms are extracted, and the resulting keywords are stored in a keyword-element map.

The second part of the indexing involves the graph itself. In the offline phase a summary graph is constructed by collapsing all class instances into their respective classes and ignoring all attribute elements. In the online phase the elements matching keywords and their edges are added back to the summary graph. Figures 1 and 2, taken from [10], show an example of an RDF graph and its summary graph, augmented with the keyword elements "2006," "Philipp Cimiano" and "AIFB."

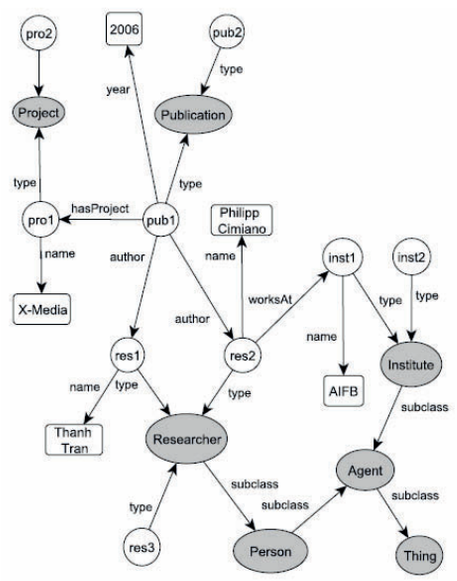


Fig. 1. RDF Data Graph

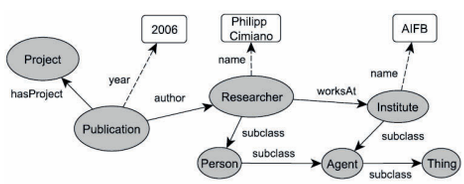


Fig. 2. Augmented Summary Graph

2.3 Finding the Minimal Matching Subgraphs

The authors of the paper used cursors tracking the travelled path. Each one contains information about the parent cursor, the keyword starting the traversal and the current path score. Since we are talking about distances between nodes, the score is actually a cost. All cursors are stored in a list sorted by cost. At each step the cursor with the lowest cost is chosen and new cursors are constructed from all of its neighbours, excluding the parent, and then stored in the list.

Once a node is found to have cursors beginning from every keyword, it is a connecting element. When such a node is found the corresponding subgraph connecting all the keyword elements, the so-called Steiner tree or graph [2], is computed and the corresponding structured query is constructed (see example in figure 3).

Since at every cycle of the algorithm only the cursor with the lowest cost is chosen, the subgraphs will also be found in descending order of cost. This means that the algorithm can stop after finding the first k connecting paths.

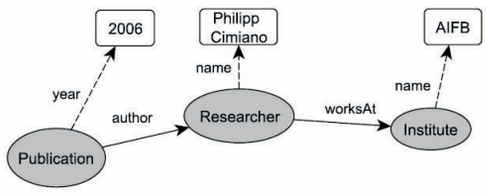


Fig. 3. Example of a Steiner graph, adapted from [10]

2.4 Scoring

In order to find the minimal matching subgraph an appropriate scoring function should be defined. As mentioned before, the score is actually a cost. Tran et. al described their function in three steps [10].

Path Length The first step is to incorporate path length into the cost. This is based on the assumption that the shorter the path between all elements is, the more related they are to each other and the resulting subgraph will fit better with the user's information need [9]. This would bring us to the first attempt at a cost function: $C_1 = \sum_{p_i \in P} \sum_{n \in p_i} 1$, where P is the set of all paths in the graph. This means that each node of the graph has the same cost.

Popularity Score In order to exploit the underlying structure of the graph, the authors of the paper proposed a second version of a cost function: $C_2 = \sum_{p_i \in P} \sum_{n \in p_i} c(n)$. In this formula, $c(n)$ represents the cost of individual graph elements. The cost is calculated as follows: $c(v) = 1 - \frac{|v_{agg}|}{|V|}$, where $|V|$ is the total number of vertices in the summary graph and $|v_{agg}|$ is the number of vertices that have been clustered to v, and $c(e) = 1 - \frac{|e_{agg}|}{|E|}$, where $|E|$ is the number of edges in the summary graph and $|e_{agg}|$ is the number of edges that have been clustered to e.

This cost calculation incorporates the popularity of each element of the graph. The more elements represented by a single node or edge in the summary graph, the higher its popularity should be, and the less it should contribute to the final cost.

Keyword Matching Score A new formula is presented for this final stage: $C_3 = \sum_{p_i \in P} \sum_{n \in p_i} \frac{c(n)}{s_m(n)}$. Here $c(n)$ is defined as before, and $s_m(n)$ is a matching

score for element n. The matching score is a function with a range between [0,1]. An element with a high score, meaning a match to a keyword, should contribute less to the cost of the path. Please note that unlike the path length and the popularity score, the keyword matching score can only be calculated online.

2.5 Evaluating Effectiveness

The authors evaluated their algorithm with the help of three different databases. Primarily, DBLP[2] was used and TAP[3] and LUBM[4] were tested in order to ensure the results are valid. DBLP is bibliographical dataset containing information about publications in computer science, TAP is an ontology with information about sports, geography, music, etc, and LUBM is a benchmark for semantic searches. To obtain query instances and ground truth, 12 people were asked to formulate keyword queries along with a natural language description of their informational need. In total, 30 queries were accumulated. The evaluation was divided into two parts, effectiveness and performance.

In this part of the study, the authors wanted to compare the three scoring functions C_1, C_2 and C_3. To assess the effectiveness of the results, the measure *Reciprocal Rank* (RR) was used. RR is defined as $RR = \frac{1}{r}$, where r is the rank of the wanted query. A query is correct if it matches the information need as defined by the natural language description. If none match RR takes the value 0.

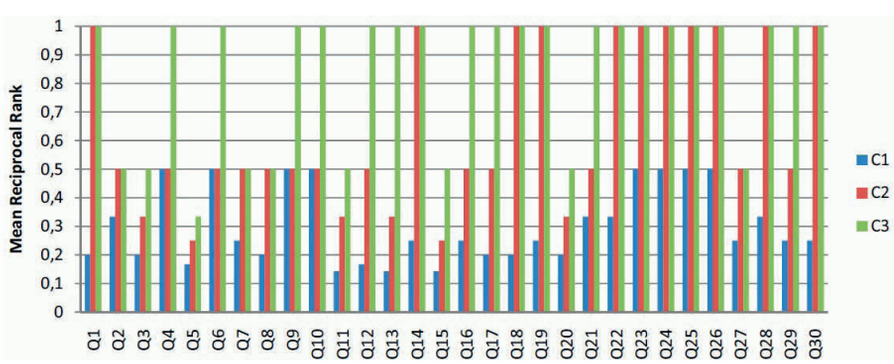


Fig. 4. MRRs of different Scoring Functions on DBLP, taken from [10]

Figure 4 shows the mean value of the RR for the queries given by the participants with respect to the DBLP database. It was tested for all three cost functions. The authors observed that when the number of alternative substructures of the graph is low, like in Q2, Q4, Q6, Q9 and Q10, function C_1 gave good

[2] http://www.informatik.uni-trier.de/ ley/db/
[3] http://www.ksl.stanford.edu/projects/TAP/
[4] http://swat.cse.lehigh.edu/projects/lubm/

results. C_2 was more effective when many substructures were involved. However, when keywords matched several graph elements resulting in ambiguity, C_2 did not perform as well. Due to this, C_3 performed the best, as it incorporated the degree to which an element matched a keyword.

2.6 Evaluating Performance

Comparative Analysis Here the authors compared the algorithm to other works, bidirectional search [4], 1000 BFS, 1000 METIS, 300 BFS and 300 METIS [3] with the help of the DBLP database. These approaches compute answers to searches, while the described algorithm computes queries. For this reason, a prototypical RDF tool called Semplore[5] was used. The authors compared query computation time and query processing time. Results can be seen in figure 5.

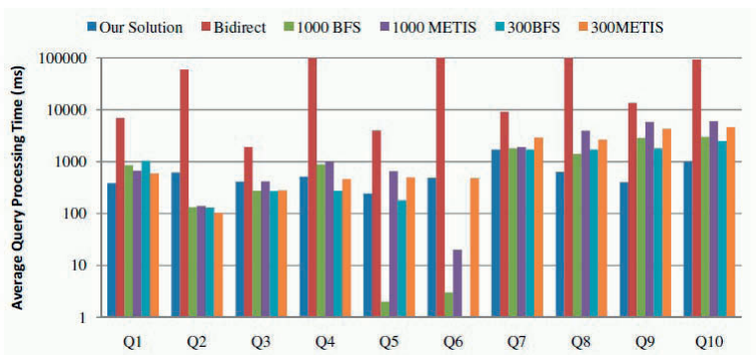


Fig. 5. Query Performance on DBLP Data, taken from [10]

It is important to mention that the authors themselves pointed out that the different approaches are not directly comparable to their algorithm, but stated: "Nevertheless, since the interaction and the number of output is the same, the comparison seems reasonable."

Search Performace A comparison was made between different choice of k with respect to query processing time. Queries of three different length were also incorporated. It is clear from figure 6a that the processing time increases linearly with k.

[5] http://apex.sjtu.edu.cn/apex/_wiki/Demos/Semplore

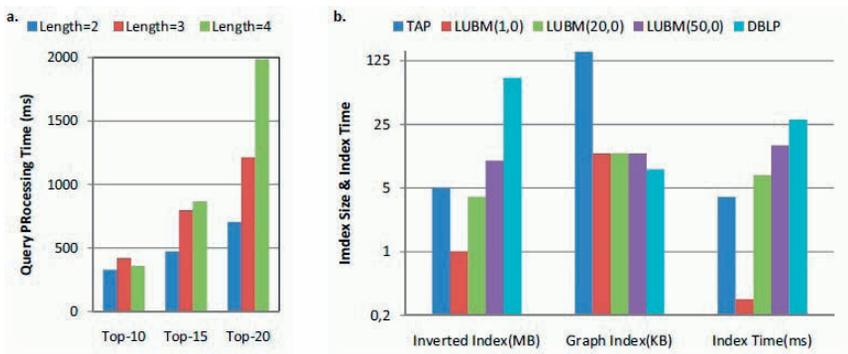


Fig. 6. a. Query Performace b. Index Performace, taken from [10]

Index Performance Figure 6b shows the impact the different databases had on the index. DBLP had many value-vertices in is data graph, and this impacted the inverted index greatly. However, TAP has more classes than other databases, and this can be seen on the graph index. The index time, according to the author, "indicates that the preprocessing is affordable for practical usage."

3 Index Structures and Top-k Join Algorithms for Native Keyword Search Databases

In their later paper, Tran et. al proposed a novel way of finding neighbourhoods of nodes in a data graph, and incorporating them in a top-k keyword join-based algorithm [5]. This algorithm will be described in this section. First, an overview will be given, then an explanation of the concept of d-length 2-hop cover, followed by the description of the algorithm and finally its evaluation.

3.1 Overview

The mentioned paper incorporates neighbourhoods of nodes in the graph, namely the keyword elements, in order to find the Steiner graphs connecting them. This time, instead of traversal through the graph, join operators are used with the help of the novel d-length 2-hop cover.

3.2 d-Length 2-Hop Cover

In order to understand the d-length 2-hop cover one first needs to understand the concept of neighbourhoods. The neighbourhood $NB_u \subseteq G$ of a node $u \in V$ is the set of all vertices and edges that are connected to u by some path. A d-neighbourhood of a node u is the set of all vertices and edges connected to u by some path of length d or less. A 2-hop cover is a path between two nodes $u, v \in V, < u, ..., w, ..., v >$, with $w \in NB_u$ and $w \in NB_v$. A d-length 2-hop cover

is then a 2-hop cover of $u, v \in V$ with path length d or less, and $NB_u, NB_v \in V$ d-neighbourhoods.

After constructing the d-length 2-hop cover for all the nodes, further pruning is done. Given $u, v \in V$ with $P_{uv} =< u, ..., w, ..., v >$ of length d or less, NB_u contains some $P_{wv} =< w, ..., v > \in NB_v$ and vice versa that are redundant. Removing P_{wv} from NB_u and P_{uw} from NB_v will result in neighbourhood that are reachable from each other through the hop node $w \in NB_u \cap NB_v$. An example can be viewed in figure 7.

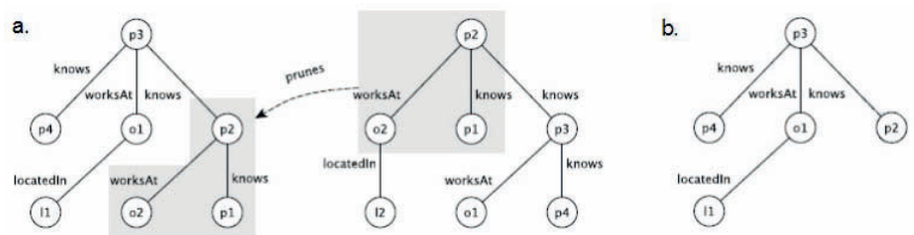


Fig. 7. a. Example of pruning, taken from [5] b. Pruned tree

3.3 Keyword Query Processing

This section describe the algorithm for obtaining the top-k Steiner-graphs, thus answering a keyword query.

Basic Join Operations The general idea of the join operations is to iteratively connect paths between the pruned d-length 2-hop covers. For this it is needed to:

1. Retrieve the neighbourhoods for each keyword.
2. Merge two neighbourhoods by performing a neighbourhood join, obtaining a keyword graph.
3. Combine a neighbourhood with the keyword graph, using a graph join.

A neighbourhood join is defined as the combination of path entries $(n_{k_1}, w), (w, n_{k_2})$ to form a keyword graph. Since the resulting graph contains only the connecting paths of n_{k_1} and n_{k_2}, during the graph join it is expanded in order to allow further connections. The final graph is then the Steiner graph. Figure 8 shows an example of a neighbourhood join, where n_{k_1} is node $l1$, n_{k_2} is node $p2$ and w is any node in the union set.

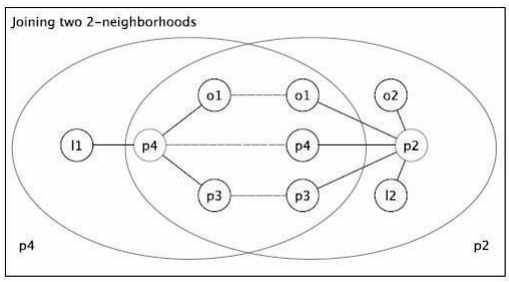


Fig. 8. Join of two neighbourhoods, taken from [5]

Integrated Query Plan A query plan is a sequence of join operations. For keyword searches longer than 2 keywords there is more than one sequence in which the join operations can be performed, each sequence can then lead to different graphs. To compute all possible Steiner graphs, all possible combinations need to be followed.

Top-k Keyword-Join Processing The algorithm to find the top-k Steiner graphs is as follows. Each join operation has a score which describes the quality of the join. The higher it is the more desired the join is. All possible joins for a particular iteration are stored in a sorted list. At each step a threshold T is set and joins with $Score \geq T$ are performed. From the resulting graphs the next possible joins are then added to the sorted list. This processes then repeats until the k-best Steiner graphs are constructed.

3.4 Evaluation

In their evaluation, Tran et. al. compared three approaches with the help of the BTC[6] and DBLP datasets. First approach is EASE [6], which is described as follows: "first all maximal 2-radius graphs, which contain at least one keyword element for every keyword, are identified. The union of these graphs computed for all query keywords is then loaded into memory, and pruned successively to obtain Steiner graphs." The other two approaches are variations of the algorithm described in this paper. First one is KJ, which includes operator rankings, and the second is KJU, which does not.

[6] http://vmlion25.deri.ie/

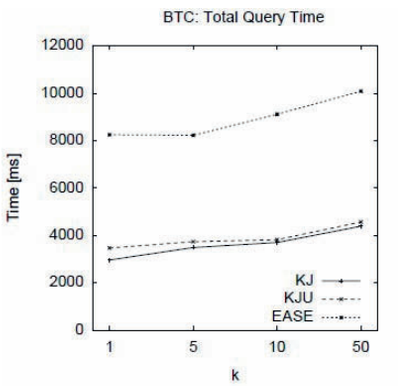


Fig. 9. Evaluation results, taken from [5]

Figure 9 shows the total query performance for the three approaches on the BTC dataset with respect to query length k. Clearly, both KJ and KJU performed better than EASE. KJ performed better than KJU on lower k's, but this difference decreases as k increases. The authors explained this phenomenon as the overhead of the operator ranking during the execution of the query. When it comes to shorter queries, though, the rankings allow for faster results.

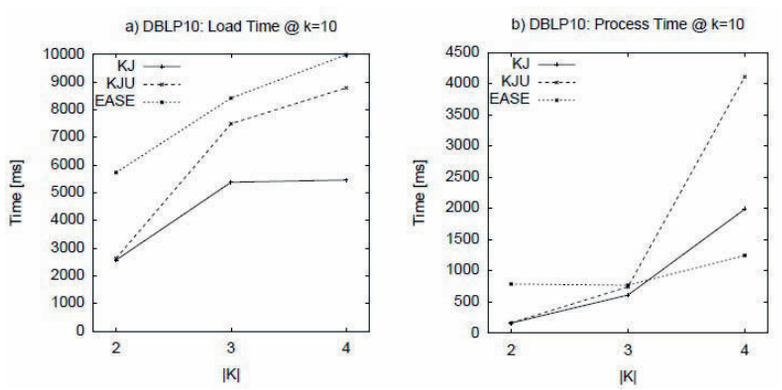


Fig. 10. Evaluation results, load and process time, taken from [5]

Figure 10 shows the access and processing time of queries according to their length, with respect to DBLP. Figure 10a illustrates the access time, and shows that ranking of join operators has a good effect on this aspect of the process, since KJ does not increase as sharply as KJU. Also visible in this graph is the evident benefits of "the more fine-grained path-level pruning implemented by KJ" which decreases the size of the neighbourhoods.

Figure 10b illustrates the processing time of a query. Here it is evident that EASE does not increase as sharply as KJ and KJU. According to the authors, this is the result of the semantic background of the proposed algorithm, as opposed to EASE, which assumes there is one central node. The benefits of operation ranking can also be shown in this graph, as the processing time of KJU increases much more sharply with the length of the query than KJ. It is evident that in most aspects, KJ is superior to EASE and that ranking of join operation has many benefits, helping make the search operation more efficient.

4 Conclusions

In this paper two retrieval methods for semantic graphs were introduced. The first method, first presented in [10], described an algorithm for finding the top-k queries using graph traversal that does not require full coverage. The algorithm stops when the first k subgraphs have been found and guarantees that these are the top-k queries according to the scoring function, which was also introduced here. This algorithm helps to predict and translate the user's information need to a semantic query that a computer would understand. This algorithm could be, however, directly compared to other methods as its output is list of queries, instead of results.

In [5], instead of loading the whole graph into memory, d-neighbourhoods of each node are calculated and stored in an index. Join operations between those neighbourhoods are scored and performed in descending order. Once the top-k subgraphs are found, the algorithm stops. This means that the graph itself does not need to be in active memory, just the list of possible joins and their scores. Since the join operations are ranked, the memory access is also cut to a minimum. This way, memory complexity goes down, which makes this method suitable for very large datasets.

Both methods presented here provide an efficient way to ease communication boundaries between user and machine. Semantic graphs such as RDF are incorporated in the translation of keywords provided by a human user to matching queries that could help identify the semantic meaning of the information need. Semantic graphs and algorithms such as the ones presented here bring us one step closer for better understanding between humans and machines.

References

1. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, S. Sudarshan *Keyword searching and browsing in databases using banks*, ICDE 2002 Pages 431-440
2. B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin *Finding top-k min-cost connected trees in databases*, ICDE, pages 836-845, IEEE, 2007.
3. H. He, H.Wang, J.Yang, P.S. Yu *Blinks: ranked keyword searches on graphs*, SIGMOID Conference 2007 Pages 305-316
4. V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, H. Karambelkar *Bidirectional expansion for keyword search on graph databases*, VLDB 2005 Pages 505-516

5. G. Ladwig, and T. Tran *structures and top-k join algorithms for native keyword search databases*, Proceedings of the 20th ACM international conference on Information and knowledge management. ACM, 2011.
6. G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou *Ease: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data*, SIGMOD Conference, 2008 Pages 903-914.
7. G. Salton, A. Wong and C.S. Yang *A Vector Space Model for Automatic Indexing*, Communications of the ACM, Issue 11, Vol. 18, 613-620, 1975
8. K. Sparck Jones *A Statistical Interpretation of Term Specifity and its Applications in Retrieval*, Journal of Documentation, Issue 1, Vol. 28, 11-21, 1972
9. T. Tran, P. Cimiano, S. Rudolph and R. Studer *Ontology-based interpretation of keywords for semantic search*, ISWC/ASWC, 2007 Pages 523-536
10. T. Tran, H. Wang, S. Rudolph, P. Cimiano *Top-k exploaration of query candidates for efficient keyword search on graph-shaped (RDF) data*, Proceedings of the 25th International Conference on Data Engineering. ICDE 2009

Part-based Features for Object Detection and Recognition

Matthias Streuber
streuber@cs.uni-kl.de

Computer Science Department, University of Technology Kaiserslautern, Germany

Abstract. Local feature detectors and descriptors are used in various tasks of computer vision. From year to year, more algorithms are presented, each of them performing better than the previous one. This paper provides a general overview of the best known algorithms and analysis them with respect to applications in document analysis.

1 Introduction

Describing a scene or an object is a common task for humans. We tend to use characteristic parts or attributes to do so. But how can a computer know, which part of an object is suitable for describing it? Algorithms for describing objects or concepts use a quite intuitive strategy: search for characteristic properties of the current scene and describe these points good enough to find them again in another image, which is maybe slightly rotated, where the perspective has changed, or the illumination altered. The question is, how to describe the scene. There are two common approaches to do so.

Global features try to describe the image in its entirety, finding a more general layout or structure by region detection or by counting the occurrence of certain color or grey-scale values in histograms. For example document analysis uses global features by comparing the overall structure of a document with already known structures for classification. Some of the early image search engines have used color histograms, to simply encode a large amount of images. By comparing these histograms, one can find more or less similar images to a reference image. The problem with global features lies in the weakness of transformation variance, image clutter and occlusion may occur. Therefore, global features are not as robust as so called local features.

Local features, also referred to as part-based features are, obviously, part of the image and as such described locally, with respect to their neighbourhood. They try to describe points of interest in an efficient way for fast and reliable comparison with a set of already known points, also called feature points or key points. These feature points should be invariant to changing effects like rotation, scaling, lighting, illumination or noise. In short, they should be repeatable. By combining these feature points, new components or feature groups can be defined. Therefore, a global feature can consist of many local features to describe a hole document or scene.

The aim of this paper is to give an overview of state-of-the-art algorithms in feature detection and description with respect to their individual properties and usefulness in document analysis, so the remainder of this paper is organized as follows. Section 2 will introduce feature properties in general, followed by a list of well known algorithms in feature detection and description task. In the next section, the presented algorithms are analysed to explain the difference in performance depending on their task. Their use and possible applications in document analysis are discussed in section 4. Finally, section 5 concludes the paper.

2 Features

As mentioned in the introduction, computer vision is mostly based on features. Depending on the analytic task, there are multiple ways for both detecting and describing these points of interest. The criteria are mostly invariant for rotation, noise, lighting and scale. Most of the time, so called corners are used for points of interest. A corner can be defined as the intersection of two edges or a sudden change of direction of an edge, but a more general way describing a corner with respect to computer vision, is a point with two dominant and different edge directions in its local neighbourhood, which can be determined by changes of the gradients, due to different neighbouring textures, colors or lighting.

The process of image analysis is mostly the same: detect points of interest by using a feature detector, reduce the amount of found points by removing points with minor local changes, encode the remaining points in an efficient and compact way as a descriptor, preferably robust to noise, rotation or changes in scale or lighting, and compare these descriptors with already known descriptors by using similarity measures like euclidean or hamming distance, depending on the type of descriptor.

In the following section, a short introduction to some state-of-the-art algorithms is given. Since some descriptors also have developed their own detector, their names will be mentioned in the detector section as well as in the descriptor section.

2.1 Detectors

As already mentioned, most of the detectors are looking for corners, since a change of gradients may be found even after a rotation, also at different scale levels. There is a way to measure the quality of a descriptor, using the repeatability. The repeatability defines the frequency of detecting feature points in image A, that are also found within ϵ pixels of the corresponding location in a transformed image B.

If not mentioned otherwise, the following detectors run on a blurred greyscale image, meaning that the image is convoluted with e.g. a Gaussian kernel to reduce the noise of the image.

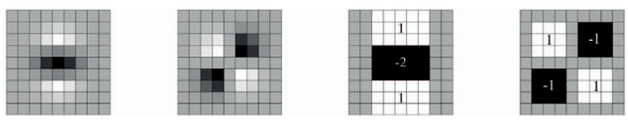


Fig. 1. Left side Gaussian second order partial derivatives in y- and xy-direction, right side the box filter approximations [2]

Harris Corner Detector

This corner detector uses the second moment matrix M (structure tensor) to detect local features in a region

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

The eigenvalues of M are then used to decide, if the neighbourhood at the current point is of interest, since two large eigenvalues indicate the presence of a corner, whereas only one large eigenvalue implies an edge. The Hessian corner detector, often referred to as DoH (determinant of the Hessian) uses the same principle, but M is a second-order matrix of the image at a point.

DoG

The Difference of Gaussian, used in SIFT [1], is simply defined by a subtraction of an image $I(x, y)$, convoluted with a Gaussian blur $G(x, y, \sigma)$ of different scale k:

$$D(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad G(x, y, k\sigma) * I(x, y)$$

Local features are then identified as local maxima/minima of D.

Fast-Hessian Detector

The Hessian detector, similar to the already introduced Harris detector, was extended by the authors of SURF [2] by the use of integral images and box filters. Rather than using the Gaussian second order partial derivatives for different parameters, the filters are approximated by box filters, see figure 1. They are much easier to compute, even more, if an integral image is used. In an integral image at each point (x_i, y_i), the sum of intensities from (x_0, y_0) (starting in the upper left corner) to (x_i, y_i) is stored, which can be calculated in linear time.

FAST

The Features from Accelerated Segment Test (FAST) stands up to its name, because it is the fastest among the others [7]. Whereas other algorithms try to detect a corner by the step wise change of the gradient, FAST simply checks

the neighbourhood of each point as follows: for a pixel p, use a circle of pixels $p_1 \dots p_16$ around p. If N of neighbouring pixel of this circle are darker (or brighter) than p with a certain threshold Θ, this point can be labelled as a corner. Both N and Θ can be set to different values to reduce the amount of found corners, but N is usually set to 12.

oFAST

Modified by the authors of ORB [5], oFAST is in contrast to its predecessor invariant to rotations. They calculate the center of mass for the current patch with respect to its intensity (intensity centroid) and store a vector from the current feature point to the center of mass as an orientation information.

2.2 Descriptors

Now, that a set of features is detected in the image, the next step is to find a way for describing these points. As mentioned in the introduction, it is always a trade-off between runtime and robustness. Storing too much information into one feature may overestimate the whole image, storing too little leads to missing robustness

SIFT

Scale invariant feature transform is one of the best known algorithms in computer vision, for detecting as well as describing features - also, it is one of the oldest. Since SIFT uses a modified DoG algorithm for detecting features, the following will only focus on the feature description for one point of interest p: p has information about its scale size from the detection step and its orientation, derived by the peak in the local gradient histogram. It should be mentioned, that for multiple peaks over a certain threshold, the original point p will be copied and handled as a new feature point with different orientation. Depending on the scale of p, a rectangular $N \times N$ grid with 4×4 subregions is laid over the image, centred in p with its orientation aligned by the maxima of the mentioned histogram. In each subregion, a local histogram of gradient directions is calculated, quantized into eight discrete directions and weighted by a Gaussian window, also centred in p (see Figure 2, left). So at the end, p is described by a $N \times N \times 8$ dimensional feature vector (see Figure 2, right). Even if the size of these descriptor vectors seems to be quite large, it is also very robust due to the amount of stored data.

SURF

Speed Up Robust Features is sometimes mentioned as the outperforming SIFT successor, since its performance with respect to runtime is much better [2]. As mentioned in the descriptor section, SURF uses Fast-Hessian as detector, but

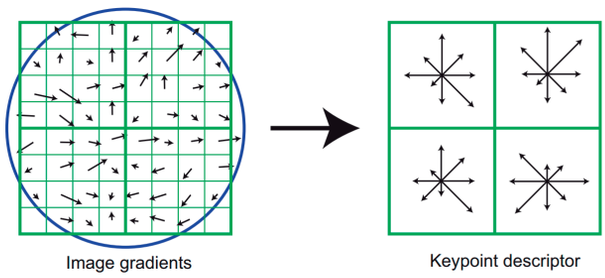


Fig. 2. The process of describing a feature in SIFT for $N = 2$ [1]

before describing the feature, SURF adds an orientation information to each feature point as follows. In a circular neighbourhood around p, the feature point, the Haar wavelet responses in horizontal and vertical direction are calculated, again by using the integral image, and afterwards weighted by a Gaussian window, also centred at p. The responses are then represented as points in a space with their corresponding strengths. The summation of the corresponding x-y-maxima in a sliding orientation window results in an orientation vector. According to the authors, rotation invariance accompanies with computational complexity. So for applications, where rotation invariance is not desired, the authors provide U-SURF, a derivative without this paragraph.

Similar to SIFT, SURF uses also a 4×4 grid, centred on p. For each subregion, simple features of 5×5 regularly distributed sample points are calculated by Wavelet responses, more precisely the Haar wavelet responses in both horizontal (d_x) and vertical (d_y) direction. For robustness against changes in intensity, the absolute value of d_x and d_y is also stored. Now, for each subregion, the responses are summed up to a four-dimensional descriptor vector $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$, resulting in a combined vector for p of length $4 \times 4 \times 4 = 64$.

ORB

Oriented FAST and Rotated BRIEF was a combination of already existing algorithms, namely FAST [7] for detecting features and BRIEF [8] for describing them. In contrast to the previously described algorithms, ORB is a binary descriptor, meaning that the resulting descriptor is not a multi-dimensional vector, but a sequence of bits. The benefit of binary descriptors lies in the comparison of two feature descriptions, e.g. for similarity measures in object detection, which can be computed by a simple XOR-operation. In BRIEF, the descriptor basically consists of pairwise intensity comparisons by a test function τ on an image patch p

$$\tau(p; x, y) = \begin{cases} 1, & \text{if } p(x) < p(y) \\ 1, & \text{otherwise} \end{cases}$$

where $p(x)$ is the pixel intensity from a Gaussian smoothed version of p at position x. The resulting bit string

$$f_n(p) = \sum_{i \leq i \leq n} 2^i \ {}^1\tau(p; x_i, y_i)$$

has a length of n, mostly 128, 256 or 512 bit, depending on the configuration. In BRIEF, the pairs x, y are sampled from an isotropic Gaussian distribution, but this may lead to a small variance and high correlation between different pairs. ORB adds a step of learning, where it takes only pairs into account with a absolute correlation over a certain threshold. To compensate the missing ability of rotation invariance of BRIEF, ORB introduces rBRIEF (Rotation-Aware Brief), a steered variation. It just takes the rotation information from oFAST, mentioned above, and rotates all binary tests at location (x_i, y_i) to the corresponding orientation of the feature point.

KAZE

This algorithm tries to avoid the Gaussian blurring step, since it does not respect the natural boundaries of objects ([3]), but reduces the amount of both noise and details. For this purpose, KAZE uses non-linear diffusion filtering in contrast to its linear alternative, mostly combined with a Gaussian kernel (see Figure 3 for comparison). For detecting the features, KAZE uses a Hessian detector and SURF for description, with slightly overlapping sub-regions.

Fig. 3. Comparison between linear diffusion (upper row) and non-linear diffusion (lower row) in ascending iterations [3]

FREAK

Fast Retina Keypoint is a more novel way of describing features, because it is motivated by the human visual system, to be exact the retina. The distribution of cones in our eye is not random, but ordered in rings of different cone density. The higher the distance to the foveal (center of the retina) the lower the acuity

of the perceived image (see Figure 4, right). This concept is combined with the one of BRISK [9]. Both have bands of circles around the feature point, where the radius of a circle shrinks with the proximity to the feature point. In FREAK, the radius changes exponentially with distance to p, also the circles overlap for more discriminant information (see Figure 4, left). Similar to ORB, this algorithm now chooses a pair of receptive fields, each of them with their corresponding Gaussian kernel, so the testing function τ becomes a one-bit Difference of Gaussian, where $p(x)$ is the intensity of the smoothed receptive field x. Similar to ORB, FREAK uses also a learning step to determine a preferable combination of receptive fields, resulting in a coarse-to-fine-pattern. So the outer rings are used for a more general classification, whereas the inner rings are used for a more detailed validation. For the comparison of features, FREAK also offers a biological motivated way, based on the saccadic search in the visual process. Only if the comparison of the first 16 bytes of the descriptor exceed a certain threshold, the rest of the descriptors will be compared, resulting in a more efficient matching step.

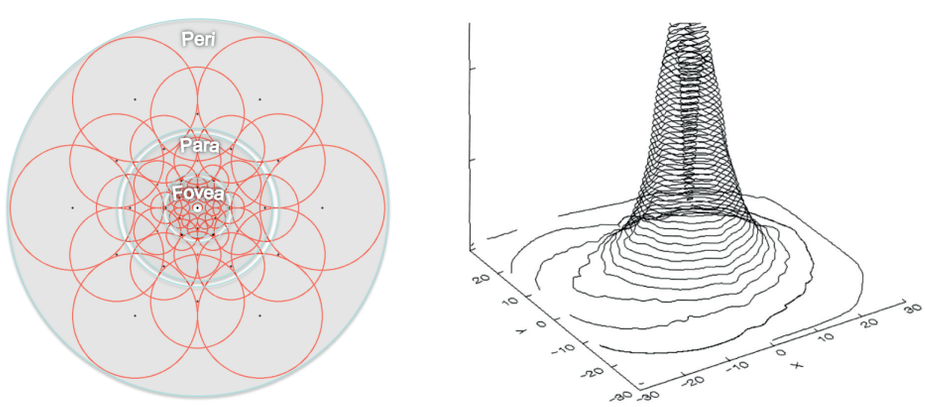


Fig. 4. Left: illustration of FREAKs sampling pattern. Right: Density of cones over the retina [4]

LIOP

Local Intensity Order Pattern use intensity in the image instead of gradients like the previously presented algorithms, which cannot handle more complex illumination changes like gamma correction, or small reflections [6]. In contrast to the other algorithms, LIOP detects points of interest by Harris- or Hessian-affine region detector, so the descriptor is working on regions rather than points. Since the detected regions may be of different size, the patch is normalized to a circular region with predefined diameter. After blurring for noise reduction, the

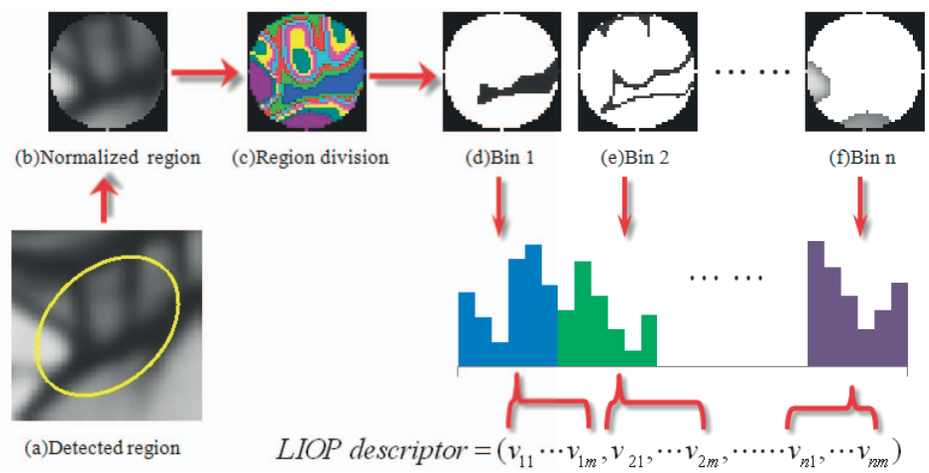


Fig. 5. Parts of the workflow of LIOP, taken from [6])

Detector	Corner	Blob	Rotation	Scale	Repeatability	Accuracy	Robustness	Efficiency
Harris	✓		✓		+++	+++	+++	++
DoG	(✓)	✓	✓	✓	++	++	++	++
Hessian		✓	✓		++	++	++	+
FAST	✓			✓	++	++	++	+++
oFAST	✓		✓	✓	++	++	++	+++

Table 1. Comparison of detectors based on [14] and [7]

patch is divided into regions of equal intensity, so called bins (see figure 5). For each point x in a bin, the LIOP is calculated as follows

$$LIOP(x) = \phi(\gamma(P(x)))$$

$P(x)$ defines a quadruplet of intensity values at locations x_1 to x_4. These points are part of a circle with fixed diameter around x, starting with the point furthest from the center of the local patch, enumerating in anti-clockwise direction. It is important to mention, because this predefined ordering of points leads to the rotation invariant feature of FREAK. $\gamma(P(x))$ returns also a quadruplet, assigning each intensity $I(x_i)$ a relative ordinal number from 1 to 4. ϕ simply maps this quadruplet by a precalculated permutation table to a unique index number q, returning a bit sequence of zeros, except the p-th bit, that will be a one. Concatenating each of the bins LIOP-histogram creates the LIOP descriptor for one local patch.

3 Analysis

As mentioned in the feature section, detectors and descriptors have to accomplish some criteria, namely distinctiveness/repeatability, robustness and (depending

Descriptor	Rotation	Scale	Viewpoint	Blur	Brightness	Runtime
SIFT	++	++	++	+	++	+
SURF	++	++	++	+	++	++
ORB	+++	+	++	++	++	+++
LIOP	++	++	+++	+++	+++	+
FREAK	+++	++	+++	++	+++	+++
KAZE	++	++	++	+++	++	+++

Table 2. Comparison of descriptors based on [10], [6], [5], [4] and [3]

on the task) compactness. All of the introduced descriptors are invariant to scale, rotation and noise. Fore a direct comparison, a unified testing scenario is needed, like in [10]. Images of a specific scene varying in point of view, scale, blur, or illumination are used to test performance and robustness. Detectors are evaluated with respect to different criteria. Repeatability describes the amount of detected features in a scene, that are also detected in another image of the same scene taken under different viewing conditions. Robustness refers to effects like blurring or noise. Table 1 gives an overview of the mentioned detectors. Most of the descriptors are using the Harris corner detector due to its good performance, but some of the novel descriptors are also using FAST to evaluate more time critical environments.

Comparing the descriptors becomes more difficult, since none of the referenced papers evaluated its algorithm against all the other descriptors. Tabel 2 tries to give a relative comparison based on this individual evaluations. In addition, table 3 from [10] gives some results of a larger testing scenario. As we can see, the binary descriptors are faster than the others, whereas LIOP tends to be the slowest due to its complexity. But with respect to precision in the matching process, LIOP is outperforming all of the other algorithms, also by matching time, based on the binary feature. Unfortunately, there are no evaluations for comparing the more recent algorithms like KAZE and FREAK with those of table 3, except for SIFT, which is also outperformed by KAZE and FREAK (see [3], [4]). Since the evaluation of table 3 heavily depends on the test set, the following section will list a few key aspects for a more general view.

Intensity. Most of the algorithms are dependant on some sort of corner detection, which in turn is dependant on significant changes of the gradients. The difficulty of detecting this information increases with lower intensity values. So, depending on the task, intensity-sensitive descriptors like LIOP may be preferred.

Detection time. For real-time applications like the tracking of moving objects, or the requirement of nominal waiting time, both, detecting and describing a feature may desire a better run time than LIOP. FAST lives up to its name, it really is a simple but relative robust feature detector. However, due to its simplicity, it is not that robust to severe changes in scale and without modification, FAST is only barely robust to rotations. But a combination of FAST as detector and FREAK as descriptor results in a robust feature system.

Descriptor	Runtime [ms]	Precision	Recall
LIOP	1801.1	0.5814	0.597
SIFT	448.6	0.525	0.533
SURF	117.1	0.485	0.513
ORB	4.2	0.448	0.470

Table 3. Runtime of descriptor computation for 1000 SURF key points. Precision and recall are calculated with SURF as detector, for a description of the testing environment, see [10]

Feature matching. The similarity measurement among two sets of features is a trade-off between the complexity of a descriptor for robustness and simplicity for faster comparison. Obviously, binary descriptors like ORB or FREAK, perform well on similarity measures, because it is a simple XOR calculation of two bit strings, combined with the calculation of the hamming distance. Motivated by FREAK, a coarse-to-fine-lookup may improve the matching process even more.

Descriptor length. Depending on the executive device, memory usage may be an argument for choosing a descriptor. SIFT, more than ten years old, is really robust to the common computer vision problems, but uses quite a lot of memory: a 128 dimensional vector of floating points. SURF reduces the amount of used memory by half, but this can still be undercut by binary descriptors like ORB.

4 Significance for document analysis

The question is, which of the introduced feature detectors and descriptors are most suitable for the tasks of document analysis. There is no hard and fast answer, since the tasks in document analysis are numerous.

One challenge in document analysis is word spotting or symbol spotting and concatenation, which are explained more in detail in [11]. For detecting symbols or letters, a corresponding reference image is needed, so one part of document analysis may be the detection of letters. Whereas letters are mostly composed of strokes, algorithms that are working on regions can be discarded. For example the only benefit of KAZE is the replacement of Gaussian blurring with a non-linear diffusion filtering. For pure letters, this seems to be kind of needless. Also the way of describing a feature must be reliable for small variations, since letters like "I" and "J", depending on the font type, share most of the feature points. ORB, that picks for a feature point p randomly neighbours to describe p, is not that reliable for finding matches in a reference set, also shown in table 3. Algorithms with a broader field of feature description, like FREAK, LIOP, SURF or SIFT may perform better for pure letter or symbol recognition.

Other challenges in document analysis are page segmentation and classification. By the knowledge of type and position of content, one can derive a possible document type for this specific page. According to [12] there are two common approaches to get this information. The first one is a top-down approach, which

starts with the hole page and divides it into smaller regions like blocks, lines or words.

The second one is a bottom-up approach. Detected components, like line fragments, are combined to characters, greater lines or blocks, depending on a closeness measure between adjacent components with respect to their size. Characters are then combined to words, lines or text blocks, relying on semantic separations like distances. The process of subdivision depends on the variance of the feature points and their spacing to each other.

As shown in [13], SIFT is already used for layout analysis. Since feature points are mostly found on or between letters, lines of text can be detected by following the highest density of feature points. All presented algorithms should be capable of doing so. There are of course smaller variations in performance, depending on the characteristics of a document. For example, some lines of text are crooked. Using a bottom-up approach may lead to minor rotation problems, when a combination of smaller skewed components is compared with a set of already known letters. A descriptor like ORB or FREAK may be more suitable in such a case. But as already mentioned, all of the introduced descriptors are more or less robust against rotation problems. It also seems to be uncommon to have an naturally blurred document image, so the advantage of descriptors like KAZE and LIOP may be redundant.

Despite the different types of document analysis tasks, one should consider the hardware requirements and the environment in general. Using for example a mobile device like a smart phone in an unpredictable environment for logo detection, properties like illumination, rotation and scale invariance are becoming more important, suggesting LIOP due to its intensity invariance. Also memory restrictions must be considered, so binary based algorithms with fast detection like FREAK or ORB may be preferred.

5 Conclusion

In this paper, a series of state-of-the-art algorithms for feature detection and description was introduced. They were analysed with respect to general problems like rotation, scale or viewpoint changes, but also concerning challenges in document analysis. It was shown, that different approaches are more suitable for certain fields of application, each of them containing some weak spots, whereas the trade-off between performance and robustness seems to lessen.

Unfortunately, most of the novel algorithms compare their performance only to SIFT and SURF, so an appropriate comparison for the remaining algorithms cannot be given, but only a analysis of ideas and motivations. Nevertheless, a tendency of more compact and faster detectors and descriptors becomes apparent, targeting the fields of mobile devices like smart phones. Also the extension of using color in the descriptors looks promising, for certain applications.

References

1. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60, 2 (2004), pp. 91-110
2. Bay, H., Ess, A., Tuytelaars, T., Gool, L. V.: SURF: Speeded Up Robust Features. Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346–359 (2008)
3. Alcantarilla, P. F., Bartoli A., Davison, A.: KAZE Features Computer Vision ECCV 2012 Lecture Notes in Computer Science Volume 7577, pp 214-227 (2012)
4. Alahi, A., Ortiz, R., Vandergheynst, P.: FREAK: Fast Retina Keypoint. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2012)
5. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: An efficient alternative to SIFT or SURF. In IEEE International Conference on Computer Vision (ICCV), (2011)
6. Wang, Z., Fan, B., Wu, F.: Local Intensity Order Pattern for Feature Description In IEEE International Conference on Computer Vision (ICCV), (2011)
7. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. European Conference on Computer Vision (ECCV), (2006)
8. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: Binary Robust Independent Elementary Features. Computer Vision - ECCV 2010, Lectures in Computer Science Volume 6314, pp 778-792, (2010)
9. Leutenegger, S., Chli, M., Siegwart, R.: BRISK: Binary Robust Invariant Scalable Keypoints. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), (2011)
10. Miksik, O., Mikolajczyk, K.: Evaluation of local detectors and descriptors for fast feature matching. 21st International Conference on Pattern Recognition (ICPR), pp 2681-2684, (2012)
11. Rusinol, M., Llados, J.: Symbol Spotting in Digital Libraries. Springer, London Limited, pp 15-47, (2010)
12. Okun, O., Doermann, D., Pietikainen, M.: Page Segmentation and Zone Classification: The State of the Art. Technical Report: LAMP-TR-036/CAR-TR-927/CS-TR-4079, University of Maryland, College Park, (1999)
13. Garz, A., Sablatnig, R., Diem, M.: Layout Analysis for Historical Manuscripts Using SIFT Features In International Conference on Document Analysis and Recognition, (2011)
14. Tuytelaars, T., Mikolajczyk, K.: Local Invariant Feature Detectors: A Survey In Foundations and Trends in Computer Graphics, vol. 3, pp 177-280, (2008)

Realizing Personal Memories with Semantic Technologies

Kristin Suchner
k_suchne@cs.uni-kl.de

Computer Science Department, University of Technology Kaiserslautern, Germany

Abstract. One major part in everyone's life is reviewing lived years. Remembering past time and dealing with it as well as creating one's own identity are essential aspects for humans. The challenge of helping people with their frequent reminiscing brought up the development of semantic technologies in memory-related content. Such contents are for instance photos, which depict special events or occasions and show aspects of one's life. Adding semantic technologies to them will support people in their reminiscing. The semantic desktop, which e.g. enables users to annotate and re-find information can serve as a first step of a semantic technology. Extending and specializing such technologies and embedding them into photos makes them a promising reminiscence tool, which will be the focus of this paper.

1 Introduction

Reminiscence is one of the most important issue in people's life. Not only does it contribute as major factor to the establishment of one's own personality, but also helps to realize and understand decisions made in the past. In addition, reminiscence helps to organize personal moments. Because of this, the desire of supporting people with their daily reminiscence rose and the need for new (or expanded) semantic technologies came into focus. One semantic technology, which can be used and further extended for reminiscence, is the semantic desktop. The invention of the semantic desktop is one way of giving people helpful additions. One example is the Personal Information Model ("PIMO") ([1]) which incorporates the mental model of users and can be integrated in OSs. Here, the basic semantic technology is the establishment of an ontology (see Figure 1).

Other similar ideas are "Semex" ([8]) and "iMeMex" ([7]). Semex (Semantic Explorer) is a system where users can directly integrate different information resources to their personal information space (which can be the data on their desktop). As PIMO, Semex uses an ontology with objects and relationships. iMeMex, on the other hand, can be integrated into standard operating systems where it facilitates data storage and analysis. All three systems use semantic technologies to support people but these aren't tools to specially assist them to reminisce.

Tools for reminiscence which use semantic technologies and help in memorizing past time or events of daily life are rarer. However, understanding people's

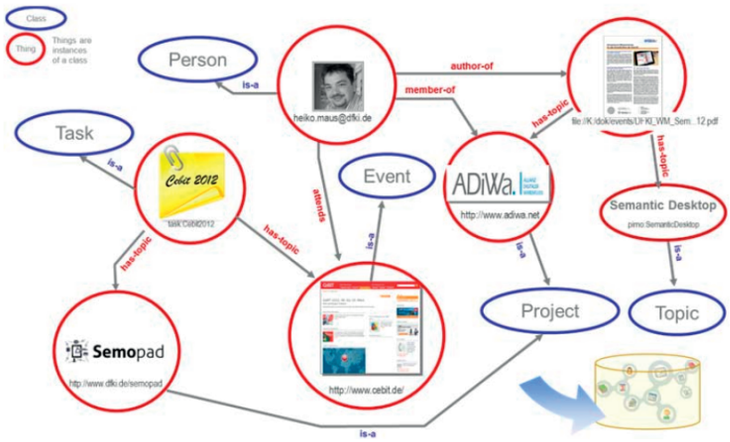


Fig. 1. Schema of a Personal Information Model (Figure taken from [1])

reminiscing technique can help creating new interfaces for user memory support. In [9], a tool called "Pensieve" is used to send users e-mails which serve as so called *memory triggers* (a reminder to reminiscence). Another idea to help people order their memories is using photos. Photos depict very important events in one's life like birth, achievements in sport or wedding. With photos one can associate stories ([5]), illustrate his or her life, remember long forgotten things or as in [6] proposed, make a photo book to have a kind of biography. Furthermore digital photos can be even more useful, they can help to collect metadata ([4]).

In this work we will first have a closer look at the PIMO as an idea of a semantic desktop, which gives a first approach of a semantic technology. Furthermore a comparison among PIMO and Semex and iMeMex will highlight different ideas of semantic technologies. After that we will discuss the meaning of reminiscence and show the idea of Pensieve with its pros and cons. Additionally we then focus on photos and illustrate how useful they can be to memorize certain events and getting metadata. Finally we will end this work with a conclusion and an own statement about the domain of personal memories in semantic technologies.

2 Representing the Mental Model of Users

As explained in the previous section we now want to take a look at tools representing the mental model of users.

Since the PIMO uses an ontology, it is an understandable system for users and therefore the semantic technology can be used further to help reminisce. The PIMO consists of concepts (person, task, event, ...), associations ("author of", "is a", "attends", ...) and associated resources (videos, e-mails, presentations, ...), which reveal the mental model of the users. Thus a common vocabulary among diverse applications is used and therefore is apprehensible.

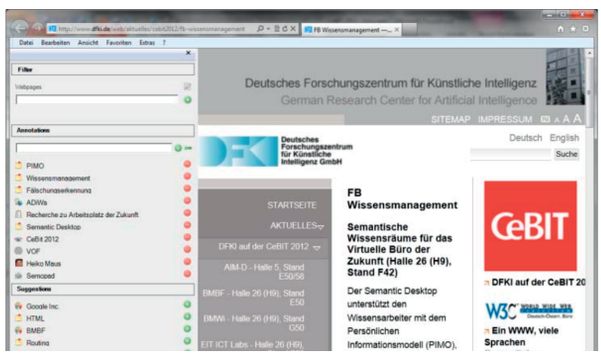


Fig. 2. MS Internet Explorer with incorporated PIMO (Figure taken from [1])

In [1] they focus on incorporating the PIMO in office applications like Thunderbird, MS Internet Explorer (see Figure 2) or Mozilla Firefox as well as in the Windows File Explorer and in task management or personal information management tools. Via plug-ins users can directly search and access as well as annotate things. That leads to the advantage of combining the resources of such applications by using the same vocabulary. This means that people no longer need to remember objects of information and their e.g. relationships but can rely on the system. The system shows coupled objects of information and supports the mental as well as physical organization of those.

As mentioned before, the PIMO reflects the mental model of the user. One approach to do so is to use the user's e-mails and search for PIMO-relevant things (as explained in the previous section, search for things like Person, Project,...). Having started with those things the PIMO finds new things, resources or relationships through the usage of PIMO incorporated office applications.

Now let's focus on some other ideas of semantic technologies which represent the mental model of users. Semex is a system where users can enlarge their own information views with public data resources as well as integrate them through aggregation of user's generated solutions. Additionally, the system helps to use already produced integration tasks (from themselves or others) to solve new problems. The concept of using someone else's solution is based on the fact that integration tasks are often reiterated several times or are nearly the same. Since reusing tasks has made work for others so much easier, Semex uses those in diverse integration tasks. This means, that the system automatically searches for such previous created tasks of others. So, the support of finding information and getting help is done by the system, which further reduces the need for users to remember objects of information they once saw. Another part of the Semex systems is, that it is based on the view that humans' minds are associative, this means, that a, as it is called in [8], "logical view" of data should be used. The logical view of data assists associations among different objects and therefore tries to, as PIMO, reveal the mental model of users (here with associations). Thanks to the logical view it is possible to link external resources to

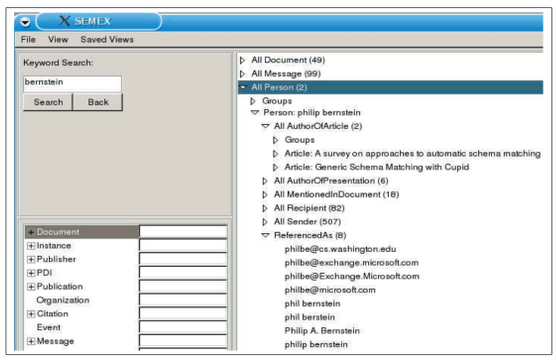


Fig. 3. The Semex system (Figure taken from [8])

one's information, which now makes it easy to help incorporating those resources to tasks which need integration. In Figure 3 there is a screenshot of Semex which *"bernstein"* and its associations are depicted in.

With iMeMex the authors want to solve the problem of the desktop being a "jungle of information". Furthermore they point out that the desktop is not a good solution for recovery, backup, versioning and synchronization, which all in all brings them to the statement that finding and storing of information (on the desktop) is too hard. A help in the organization of user's mental model is given here. The iMeMex architecture therefore has a layer which supports storage and analysis for different data and should replace the current file system. More precisely, iMeMex takes the home directory folder and out of it creates a *"iMeMex folder"* with which iMeMex gains full control. iMeMex integrates itself into existing OSs and so no more import/export or synchronization is needed. Here not only a semantic help is supported, like in PIMO or Semex, but a whole new addition to the operating system is done. iMeMex organizes the information on the e.g. desktop and supports people in their need of structuring content. The iMeMex folder (see Figure 4) consists of *physical resources* (those which were put in by the user) and *virtual resources* (those which were added by the system). Virtual resources can be metadata, more structured sights of the physical resources, the kick-off for a search task or links to similar items, and so the virtual resources build the semantic and associative part of the system which makes it a competitor for Semex and the PIMO.

In this section, we explained three tools which use semantic technologies to help people remembering, organizing or structuring as well as searching, re-finding and annotating objects of information. They support people in reducing their effort to find associated objects and generally it easier to deal with a huge amount of information. However, none of them uses the semantic technology to support people in their most occupying task, reminiscing.

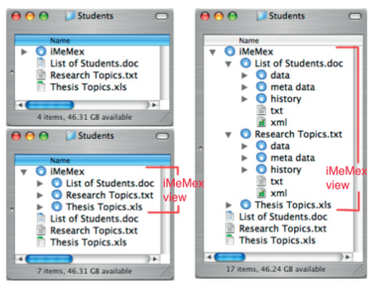


Fig. 4. Three iMeMex views (Figure taken from [7]). Top-left: folder includes "List of Students.doc', "Research Topics.txt', "Thesis Topics.xls' plus a subfolder "iMeMex" (virtual resource). Bottom-left shows the view on the expanded iMeMex folder (for each physical resource one subfolder added), whereas the right view shows the expansion in it for two subfolders (containing metadata and further information).

3 Reminiscence and the Tool Pensieve

In the next sections we will lay the focus on tools which use semantic technologies to assist people reminiscing. Before describing these tools and discussing the pros and cons we first explain what reminiscence is and why it is so important in this subject.

As mentioned before, reminiscence supports people in building their own personality and helps to cope with past situations and experiences to focus on problem solving. Furthermore, we obtain relationships with reminiscence. Reminiscence is spontaneous and often happens by chance. Also, important events or traditions of culture or just thinking build reminiscence. In this work we want to focus on technologies which support personal memories and thus, reminiscence is a key to do so.

In ([9]) they focus on everyday reminiscence and create a tool which helps to do so. Pensieve is using the supposition, that topics in social media support reminiscence. Status updates, status messages or posts (on MySpace[1] or Facebook[2]) comprise daily situations and might be of interest in the future, but unfortunately as new content overwrites old ones, they are buried in oblivion. To see how good social media content really is, the authors wrote a Facebook app and made a study with 96 students who were supposed to rate on a 1("extremely unlikely") to 5("extremely likely")-point Likert scale whether or not posts made them reminiscing. The result was a median of 3.0 with a standard deviation of 0.7 which clearly showed that there was no unanimously voting for or against the hypothesis that such content was good or bad. Most of the time it depended on whether people easily started reminiscing or not. A better statement was given on which content reminiscing helped. Content about people they wanted to think of, very precise descriptions of e.g. events or unknown or rich experiences were

[1] https://myspace.com/
[2] https://www.facebook.com

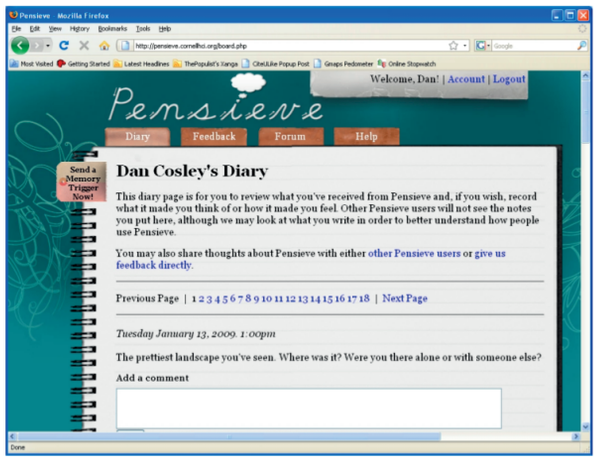


Fig. 5. The Pensieve website with the diary area (sended memory trigger included) (Figure taken from [3])

declared as precious. Since several people told the authors that they liked the study and enjoyed reminiscing about older facts the overall statement in [9] is, that social media content supports reminiscence and encourages them to build a tool helping people to reminisce. Pensieve is a tool which is "integrated in everyday life" and not an extra and separate but rather an additional tool. Through its website[3] people can create their own account and if wanted connect it to other ones (social media accounts). In addition the Pensieve account is connected to a Pensieve service which consists of text prompts (like "What do birthdays look like in your family?"), in which people can choose how often they want to get these memory triggers. The main contact with Pensieve is via email, memory triggers are picked from an item by chance (from user's services) and then are sent to the user through an email which contains exactly this item, which shall bring users to reminisce (see Figure 5). In addition, Pensieve allows to write a diary. For that, users have the opportunity to apply to the emailed memory triggers via writing a story for the diary. The created diary includes the item of the sent trigger and the user's reply to it. They asked people in their evaluation if they used Pensieve in their daily life and if it served the goal of helping people to reminisce. Most people interacted with Pensieve in a spontaneous way, which is reflected by the fact that the users concluded to prompts in a time variance of max one hour. Furthermore, some people also commented on using Pensieve daily and writing diaries. The final statement was, that people really liked being supported in reminiscing, they liked the thinking and writing part and it served to lighten their mood, not forgetting past events made them happy.

[3] http://cornellhci.org/pensieve

4 Support via Photos

Having an impression of how important reminiscing is and showing one tool, called Pensieve, which helps doing so, we now want to explain what other concepts can be used to support people in their daily memory overload. Photos are a perfect tool to remind people of events, special days or shared moments with loved ones. People don't need to intensely think about the past, the people and places they went to, they just have to look at pictures and instantly a memory pops up. This makes them suitable for reminiscence. Because of that, the next sections will present in which ways photos can be used to serve as assistance and are able to reflect personal memories.

4.1 Pensieve with Photos

We have already talked about Pensieve's usage of emailed memory text triggers. But Pensieve sends texts as well as photos. In this scenario the emailed prompts are not a text but a photo, people still can reply to it and create a diary where the photo is now used as the reminder to reminiscence. In their paper ([9]) they found out, that personalized photo triggers suited better for writing a diary. People liked having a visual clue to their memories and therefore used to write diary entries more often after having a photo prompt. However, such entries were sometimes rather story telling than writing about the content of the picture (where was it, who is seen, what time, ...). Therefore the usage of both, text and photo triggers, makes the system more valuable to users.

4.2 MyLife Photo Book

Since writing a diary is for oneself but reminiscence and especially people's personal memories are related to other persons, it is better to use a photo book to communicate one's own life (or special events) to e.g. family. A variety of photos and not only one single photo will increase people's reminiscing.

In [6] they take photos to support people by biography writing, because they make it easier (particular at the end of life) to remember the past and one's heritage. Their approach contains three phases, first the "collection" phase in which they gather important information. This means in detail, data about the persons life like "born in", "grew up in", the attended schools and so forth which can be done by him- or herself or even if the person is already dead by relatives. With that a common vocabulary is created and builds the semantic technology in the system. This step is important because the person has already to think about his or her past years. It also brings up memories for relatives (in Figure 6 the information gathering interface can be seen). A second step, the "interaction" step, starts with a first framework of the photo book, using the facts collected in the first step. It fills the pages with those details and searches for further information (from e.g. Wikipedia[4]), which can be additions according

[4] https://www.wikipedia.org/

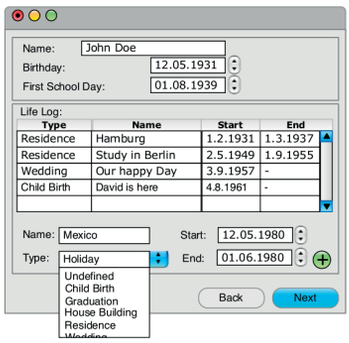


Fig. 6. Gathering information via prototype (Figure taken from [6])

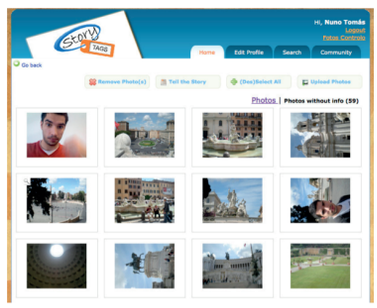


Fig. 7. The website (Figure taken from [5])

to location (completion in form of maps, pictures with historical backgound, ...) or time (e.g. completion in form of events at that time). Hence, after the second step a first version of the photo book is created which contains the person's way of life. In the last step, the so called "remembrance" step, the user now begins to tell his story by adding material to fill in the blank. The user adds photos to the predefined gaps, information about persons on pictures, some other details he knows, a story to the event or whatever he wishes to add. Finishing the last step leads the user to his own "MyLife Photo Book".

4.3 A Narrative Based Interface

Writing a biography and thus trying to remember one's own life is a good way to get some of one's own personal memory overload captured. But this work is often done by the elder generation and not every memory is appropriate for a biography but still worth remembering. Furthermore, it doesn't help to re-find a special memory out of many data. Since everybody (old or young) reminisces, another idea including photos will be presented now.

In this subsection an interface which allows users to tag their media with associated stories to it will be discussed. According to [5] problems like ambiguity and interchangeability appear in normal tagging (as it is used in e.g. Flickr[5]) but can be avoided using stories. Another problem is that not everyone uses the same tag for the same object and therefore reusing normal tags is not sufficient.

Their approach is to get stories to media from users so that the factor of reuse will increase. Before creating their narrative-based interface, they interviewed twenty people about how they described photos with stories. They wanted to see which elements of stories were used more often than others and they were interested in whether stories have a specific structure and use rules. To get that information, the questioned people had to pick three personal pictures and tell their story. The authors were interested in the elements: Time, location, author,

[5] https://www.flickr.com/

purpose, photo type, size, event, device, description, people and quality. Furthermore they wanted to see which elements have to be asked for and which ones were told automatically. The results showed that time and location were used the most, so people easily associated these facts with pictures. Device, author, type and dimension were the elements which needed to be asked for, where author and device were the two of the four, which after asking, were often remembered (the other two were nearly forgotten by all users). This indicates that the first two are in fact useful pieces of information but not as easily remembered as time and location. Having the information about the important elements the question about structure was analysed. They noticed, that these elements were in a relation among each other. When people talked about the device they also remembered the quality, when they knew the time they also talked about the location and when they remembered the location they thought about the event. This shows that they used an ontology as semantic technology, since they searched for important elements and their relationship among each other. With that knowledge they built an interface which was used for storytelling. Re-finding a photo out of all the pictures uploaded to the interface is pretty simple, users enter an indication term out of the elements described earlier. According to a "story-difference" metric the similarity of elements included in stories and the search term is calculated, the best matches are filtered out and are shown (the metric takes frequency and structure into account). In Figure 7 the interface is depicted (which was used for the study of story telling). In the study the users were presented with the elements (discussed earlier) in turn and could choose one of the answers collected by the interviewers, write a text (story to the picture) or choose "can't remember"(if the element is not important for the story of the picture). To confirm their hypothesis that the narrative interface is better than another one with normal tagging, they did a second interface for the latter one. They found out that in the narrative system, the stories used 7.96 elements on average whereas in the normal tagging system the average was 1.35 tags (here tags and elements are equal). This shows that users can commit 5.9 times more information to photos when using the narrative form. Additionally they found out that 94 percent of the elements were reused whereas in contrast only 36 percent in the normal tagging system.

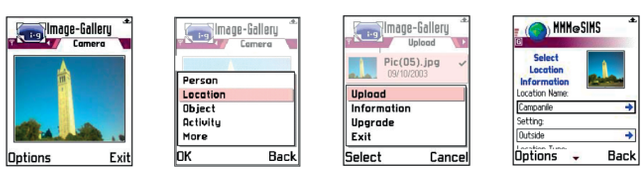


Fig. 8. MMM Image-Gallery (Figure taken from [4])

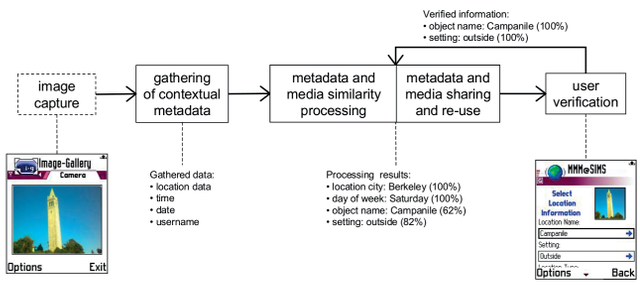


Fig. 9. Information gathering (Figure taken from [4])

4.4 MMM ("Mobile Media Metadata")

After talking about adding stories to photos to connect as much information to them as possible and to make it easier to find them again we now concentrate on a system which automatically generates metadata out of pictures taken by a phone. As we have already seen semantic technologies in photos, like common vocabulary and ontologies, we will now see an interface which also uses metadata as semantic contribution to the system. The system should help people to organize their pictures without the effort of looking through them and manually adding information. Here the help is related to minimize people's time on remembering content and annotating it. In [2] they illustrate that using mobile devices is very intelligent in that way, that the device already gives information about where it is and at what time. Furthermore information sharing and collaboration can lower annotation efforts even more and, on the other hand, increase the amount of semantic data like who is in the picture, what is he doing or what building is in it.

That is why they established a framework called MMM, which allows annotations at the actual snapshot time. It is based on a client server architecture, where the client has two components. First the "Image-Gallery" (see Figure 8), which collects location metadata as well as time and date (all taken through GSM). After having this metadata it is sent to the server. Via a collaborative repository of photos (already annotated ones) the client and server can interact and therefore automate the annotating step. Matches between the new uploaded photo and the photos in the repository make it possible to do "guesses", which can be marked as correct or be edited by the user (second component). The guesses are presented through a drop down list (best one on top according to the server's similarity algorithm), and by clicking they can be confirmed as correct or if wrong simply be corrected with another text. Finding out the semantic context, e.g. who is in the picture, is done by weighting the information known by the user who took the picture. With that information the system can look at previously shot pictures and see who was depicted in them and furthermore take into account the frequency of annotations. It can guess with a certain probability who might be in the picture. The same idea can be used by guessing the activity. Already annotated old pictures similar to the new one might illustrate the

same activity with some certainty. The guessed elements are based on ontology elements, like activity or place, and integrate a second semantic component to the system. The whole process can be seen in Figure 9.

5 Conclusion

In this work we have discussed the topic of supporting personal memories in terms of organizing, managing and remembering them. The main reason for the importance of this subject is the huge amount of time which users take for reminiscing. We first talked about semantic technologies which can be used further and extended for reminiscence. The PIMO, as one of those, is a representative of a semantic desktop which helps users incorporating their mental model in their tasks. After comparing the PIMO to other systems and discussing the used semantic technologies, we switched to tools which take (and extend) these ideas to support reminiscence. In the beginning we highlighted how important reminiscence in the area of personal memories is. Focusing on promising objects to support personal memories, we picked photos as an example of being a helpful item to remember and re-find memories as well as organize them. We finished the topic through introducing different systems and showing how they tried to assist people in their everyday reminiscing.

Since reminiscing is a major component for building one's own identity, establishing and maintaining relationships, dealing with past situations or just remembering wonderful moments in one's life, I think developments of tools are needed to support people with these aspects. I found a lot of good ideas solving the problem to help people remembering, organizing, structuring or searching, re-finding and annotating objects of information. The proposed tools, in general, make it simpler to deal with a huge amount of information. In my opinion, those systems are not only good sounding suggestions but they are kinds of implementation which actually assist users with completing tasks. The semantic desktop as an example chosen here, enables people to manage content in a humanly logical way, and makes it easier and even faster to search and organize their objects of information. Unfortunately I see much more need for further research in the personal memory area and in coping with daily life situations, so more investigation for tools which effectively support reminiscence. The presented solutions are often very specific, e.g. writing a diary sounds good in the first place but after thinking about it reveals still a lot of effort by the users themselves (like writing stories or correcting the system). In my point of view, they build a good baseline for future work but definitely more research has to be done. People need to be more than just supported, they also require to be pushed to reminisce so that the positive aspects of it, like personality forming or coping with past events, are developed as much as possible.

References

1. Heiko Maus and Sven Schwarz and Andreas Dengel.: Weaving Personal Knowledge Spaces Into Office Applications. Proceedings of 5th International Conference on

Integrated Systems, Design and Technology (ISDT 2012)
2. Wilhelm, Anita and Takhteyev, Yuri and Sarvas, Risto and Van House, Nancy and Davis, Marc: Photo Annotation on a Camera Phone. CHI '04 Extended Abstracts on Human Factors in Computing Systems (2004)
3. Cosley, Dan and Akey, Kathy and Alson, Brian and Baxter, Jonathan and Broomfield, Mark and Lee, Soyoung and Sarabu, Chethan: Using Technologies to Support Reminiscence. Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology (2009)
4. Sarvas, Risto and Herrarte, Erick and Wilhelm, Anita and Davis, Marc: Metadata Creation System for Mobile Images. Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services (2004)
5. Tomás and Nuno and Guerreiro and Tiago and Jorge and Joaquim A. and Gonçalves, Daniel: A Narrative-based Alternative to Tagging. Proceedings of the 21st ACM Conference on Hypertext and Hypermedia (2010)
6. Philipp Sandhaus and Hannah Baumgartner and Jochen Meyer and Susanne Boll: That was My Life: Creating Personal Chronicles at the End of Life. Pervasive '10: Proceedings of the Eighth International Conference on Pervasive Computing (2010)
7. Dittrich, Jens-Peter and Salles, Marcos Antonio Vaz and Kossmann, Donald and Blunschi, Lukas: iMeMex: escapes from the personal information jungle. VLDB '05: Proceedings of the 31st International Conference on Very large data bases (2005)
8. Xin Dong and Alon Halevy and Ema Nemes and Stephan B. Sigurdsson and Pedro Domingos: Semex: Toward on-the-fly personal information integration. Workshop on Information Integration on the Web (IIWEB 2004) (2004)
9. Cosley, Dan and Sosik, Victoria Schwanda and Schultz, Johnathon and Peesapati, S. Tejaswi and Lee, Soyoung: Experiences With Designing Tools for Everyday Reminiscing. Human-Computer Interaction, Vol. 27, No. 1-2 (2012)

German Research Center for Artificial Intelligence (DFKI) GmbH

DFKI Bremen
Robert-Hooke-Straße 1
28359 Bremen
Germany
Phone: +49 421 178 45 0
Fax: +49 421 178 45 4150

DFKI Saarbrücken
Stuhlsatzenhausweg 3
Campus D3 2
66123 Saarbrücken
Germany
Phone: +49 681 875 75 0
Fax: +49 681 857 75 5341

DFKI Kaiserslautern
Trippstadter Straße 122
67608 Kaiserslautern
Germany
Phone: +49 631 205 75 0
Fax: +49 631 205 75 5030

DFKI Projektbüro Berlin
Alt-Moabit 91c
10559 Berlin
Germany
Phone: +49 30 238 95 0

E-mail:
reports@dfki.de

Further information:
http://www.dfki.de