ForgetIT

www.forgetit-project.eu

ForgetIT

Concise Preservation by Combining Managed Forgetting and Contextualized Remembering

Grant Agreement No. 600826

Deliverable D6.4

Work-package	WP6: Contextualization / Decontextualization
Deliverable	D6.4: Contextualisation Framework and Evaluation
Deliverable Leader	Mark A. Greenwood
Quality Assessor	Robert Logie
Dissemination level	Public
Delivery date in Annex I	31st January 2016
Actual delivery date	22nd March 2016
Revisions	7
Status	Final
Keywords	context, evolution, text, images, preservation

Disclaimer

Revision History

Date	Version	Major changes	Authors
17/11/2015	0.1	Initial Version	USFD
01/12/2015	0.2	First draft of image contextualization	CERTH
17/12/2015	0.3	First draft of application examples First draft of text contextualization First draft of context evolution	L3S, USFD L3S USFD, L3S
21/01/2016	0.4	First draft of Wikidata approach	DFKI
29/02/2016	0.5	Edited for consistency	USFD
03/03/2016	0.6	Better referencing of the usecase examples	DFKI, USFD
22/03/2016	1	Final version after QA	USFD

List of Authors

Partner Acronym	Authors
CERTH	Vassilios Solachidis, Olga Papadopoulou, Konstantinos Apostolidis, Damianos Galanopoulos, Dimosthenis Tastzoglou, Vasileios Mezaris
DFKI	Bahaa Eldesouky, Heiko Maus
L3S	Nam Khanh Tran, Christoph Hube, Claudia Niederée
USFD	Mark A. Greenwood, Johann Petrak, Genevieve Gorrell

Table of Contents

Executive Summary **7**

1 Introduction **8**

1.1 Target Audience 8

1.2 Structure of the Deliverable 8

2 The Big Picture **9**

2.1 An Illustrative Example 9

2.2 Technical Discussion 13

3 Text Contextualization **15**

3.1 Contextualization for Microblog Topics Using Wikipedia Temporal Information 15

3.1.1 Problem Statement 15

3.1.2 ForgetIT Approach 16

3.1.3 Framework 17

3.1.4 Experiments 19

3.2 Time-travel Translator: Automatically Contextualizing News Articles 23

3.2.1 Overview 23

3.2.2 Time-aware Contextualization 24

3.2.3 Demonstration of The System 26

3.3 Text contextualization using Wikidata 29

3.3.1 What is Wikidata? 29

3.3.2 Wikidata Characteristics 29

3.3.3 Wikidata's APIs 30

3.3.4 Implementation 32

3.3.5 Conclusion 33

4 Image Contextualization **34**

4.1 Problem Statement 34
4.2 ForgetIT Approach 34
4.2.1 Advances in Comparison to the Previous Version 37
4.3 Experimental Evaluation and Comparison 37
4.4 Software Implementation and Integration 39

5 Context Evolution 42
5.1 Structured Context Evolution 43
5.1.1 Problem Statement 43
5.1.2 ForgetIT Approach 43
5.2 Unstructured Context Evolution 46
5.2.1 Problem Statement 46
5.2.2 ForgetIT Approach 46

6 Context in Applications 50
6.1 Forgetful Search 50
6.2 Situation Search 52
6.3 Context Aware Search 53
6.4 Context Evolution 55
6.4.1 Introduction 56
6.4.2 Models and definitions 56
6.4.3 Wikipedia as a Contextual Graph 58

7 Conclusions 60
7.1 Summary and Conclusions 60
7.2 Assessment of Performance Indicators 60
7.2.1 Conceptual Framework for Contextualization 60
7.2.2 Contextualization Tools 60
7.2.3 Techniques for Context Evolution 61
7.3 Lessons Learned 61

7.4 Vision for the Future 61

8 References **62**

Glossary **65**

Executive summary

The goal of WP6 was to develop methods that enable the contextualization of both text and images. This is intended to support the future understanding and re-use of preserved documents by augmenting them with details of the context surrounding them at the time of creation.

In this deliverable we present the third and final release of the ForgetIT techniques for contextualization, which build upon those reported in the previous deliverables of this work-package [Greenwood et al., 2014, Greenwood et al., 2015], along with evaluation results for each component. We also present an updated approach to contextualization which focuses on a user centred approach.

A final discussion section highlights how the ForgetIT approach is mirrored by the reported components and sets out how the components have fulfilled the expected outcomes documented in the Description of Work as well as setting out a path for future work, beyond the project, in this area.

1 Introduction

This deliverable documents the third and final release of the ForgetIT components for contextualization. These components have been developed following a thorough state-of-the-art review [Greenwood et al., 2013] to carry out the numerous tasks defined within the ForgetIT approach to contextualization. The ForgetIT approach was initially documented via a formalism heavy description [Greenwood et al., 2014] that, in retrospect, was difficult to follow for those not heavily immersed in the project. A more user centred description is included in this deliverable which is an extended version of that originally published in [Greenwood et al., 2015].

It is important to note that the discussions of contextualization approaches reported in previous deliverables [Greenwood et al., 2014, Greenwood et al., 2015], which have not altered significantly, are not repeated in this deliverable although they are still available for use within the PoF framework. We invite interested readers to revisit these previous deliverables for further details.

1.1 Target Audience

As this is a prototype deliverable it is, by it's very nature, quite technical in places. Each section is, however, self-contained and aimed at a specific audience and as such relevant parts can be read by the appropriate audience. Also each section starts with a broad overview which should allow a basic understanding of the component being described even if the technical details are outside the area of expertise of the reader.

1.2 Structure of the Deliverable

The remainder of this deliverable is structured as follows. Firstly we present an updated overview of the ForgetIT approach to contextualization that is based around a worked example as seen from a user's point of view. Following on from this high level discussion there are three sections describing components for image contextualization, text contextualization, and context evolution. This is followed by a section which documents a number of ways in which context can be used within a wider application. The deliverable ends with a discussion section.

2 The Big Picture

In this section we look at how both general and personal world knowledge can be used to contextualize documents (both text and images) and how this additional information can be preserved and utilized. We specifically focus on personal world knowledge as even in the far distant future today's world knowledge should, barring a large scale *digital dark age* [Kuny, 1998], be accessible but its use is limited as it only extends to well known or *famous* entities and events. Everyday personal details, events and relationships form a rich context for understanding documents and this must be captured as part of any successful approach to contextualization.

The rest of this section is split into two main parts. Firstly there is a worked example showing the different sources of information available and how these link together to form context. This is followed by a more technical discussion that explains how such a contextualization approach works within the confines of the ForgetIT project.

2.1 An Illustrative Example

Rather than focusing on the technical details we shall start by looking at an example which we hope will illustrate the main concepts and which paved the way for the development of many of the components documented in the rest of this deliverable. This example revolves around a diary entry and a photo covering a single event that occurred during a ForgetIT consortium meeting held in Luleå, Sweden.

Jörgen had offered to take Elaine, Maria, and Robert out to Gammelstad this morning before the meeting started as they are looking at running a memory study there over the summer. When Jörgen arrived to pick them up I decided to be cheeky and ask if there was room in the car for one more. There was so I got to do the touristy thing of looking around while everyone else did some actual work. A large snow pile made an excellent back drop for Robert to take a group photo which hopefully I'll get a copy of at some point. It was certainly an interesting place to look around and you can understand why it is a UNESCO world heritage site.

Diary Entry 2.1: File metadata lists the author as 'Mark A. Greenwood' and the creation data as the 12th of February 2014

If we assume that the diary entry (shown on page 9) is the first document (text or image) that we are preserving then we will have no existing personal store of world knowledge to call upon. The first stage of contextualization will therefore be to link the document to a source of general world knowledge and for this example we will assume this is Wikipedia[1].

For this example diary entry links would be generated to the pages for Gammelstad[2] and

[1] Implementations of this idea are more likely to use DBpedia but for a *hand worked* example Wikipedia makes more sense. See the technical discussion on page 13 for more details.

[2] `http://en.wikipedia.org/wiki/Gammelstad_Church_Town`

Photo 2.1: Metadata associated with this photo show that it was taken at 7:48:33 UTC on the 12th of February 2014 at N65°38′42.75″ **E**22°1′38.573″ **at a magnetic bearing of** 171.5°**.**

UNESCO[3] as they are the two *famous* entities within the text. This leaves five entity mentions which, if not well known, must fall within the users personal world knowledge: Jörgen, Elaine, Maria, Robert, and I.

With no pre-existing personal world knowledge the only information we have is the file metadata which allows us to map the first person pronoun *I* to the document's author *Mark A. Greenwood*. This leaves us with four unknown people for which the system would need to prompt the user for additional data. At a minimum this additional data should probably consist of a person's full name and their relationship to the user (i.e. Jörgen Nilsson is a collaborator on the ForgetIT project and works at the Luleå University of Technology).

Alternatively, such information could be available from the user's own information sources, such as (electronic) address book, e-mail account, or calendar. This is in fact the approach taken in WP9 with the Semantic Desktop: mining a user's information sources built from his or her Personal Information Management (PIM) activities and generating personal world

[3] http://en.wikipedia.org/wiki/UNESCO

knowledge in the Personal Information Model (PIMO) (see also D9.3 [Maus et al., 2014] Section 2). This information can then be used by accessing the PIMO to generate possible options in the above example. Even when a user starts from an empty PIMO, steps can be taken to bootstrap the process such as crawling available PIM sources to identify useful structured data. For instance, crawling the e-mail account of Mark and identifying contacts, groups, and projects (as presented in [Schwarz et al., 2011]) would produce a list of people who worked on the ForgetIT project during the time the picture was taken. Furthermore, the Personal Preservation Pilot shows how the PIMO Diary can be used for generating even richer entries from the information available by combining more PIM activities such as calendar, email, or notes (see D9.3 [Maus et al., 2014] Section 2.5.2).

Having gathered this information the next step would be to store the current personal world knowledge in the archive. The diary entry could then be contextualized by storing not only the entry itself but links to both the general and personal world knowledge within a submission information package (SIP); the links to the personal world knowledge being with reference to the archived information package (AIP) containing the most recent archived version.

Not only does this process allow us to store extra context information alongside the diary entry, but it has started the process of building up a repository of personal world knowledge which can in turn be used to contextualize new documents more accurately and with less user intervention. It would be beneficial if this linking process was part of an interactive feedback loop [Goetz, 2011] so that users could see the benefit of existing data being used to enhance their output (i.e links to Gammelstad and UNESCO appearing *magically*) as this would encourage them to provide personal data when prompted. Within WP6 for instance, this is achieved by providing with Seed (see also Section 3.3) a means to contextualize texts as early as during writing the text. Existing personal (such as the PIMO) as well as general world knowledge (such as DBpedia) is recognized and proposed to the user or even directly annotated if adequate.

Having fully contextualized the diary entry, let us turn our attention to the task of contextualizing and preserving Photo 2.1.

The first thing to note is that the photo and the diary entry clearly act as context for one another. If both were archived at the same time as part of the same submission information package (SIP) this context would be clear but as we are assuming that the photo is being preserved at a later date than the diary entry then this link needs to be made clear. This highlights the fact that one important source of contextual information are the documents which have already been archived. Previously archived documents can act as context in two ways:

- A SIP can explicitly reference archived information packages (AIP) as context. In this case the SIP containing Photo 2.1 would explicitly reference the AIP containing Diary Entry 2.1 to provide contextual information that explains both the occasion of the photo and its content (i.e. the people and location).

- The processing of each new SIP adds to what we know about the individual users

Photo 2.2: A *wider* context?

(where user could be a company not just a person) personal world knowledge. In this example contextualizing the diary entry will have led to four previously unknown people being added to the user's personal world knowledge.

It is likely that the explicit linking of the diary and photo would be a manual action taken by the user (or being supported by an application such as the PIMO Photo organization app presented in D9.3). It may, however, be possible to suggest the relationship to the user based on the associated metadata and context information generated when the diary entry was preserved. Firstly the photo and the diary were both created, according to the metadata, on the same day, the 12th of February 2014, which at least suggests a common context. Furthermore the GPS information can be used to search for Wikipedia pages describing nearby places which would link the photo linked to the same page describing Gammelstad[4] as used to provide general world knowledge for the diary entry.

The GPS metadata associated with the photo could also be used to select other photos which could act as context. In this example, Photo 2.2 has almost identical GPS metadata but clearly shows a wider view than the original photo and would help to provide a larger visual context. Furthermore, GPS metadata can be used (e.g. employing Google places API) in order to list the places that are located close to the image GPS coordinates. Also, combined with magnetic bearing (if available) and the Focal length of the shot, the subset of the places that may be visible can be extracted. An example of place info for Photo 1 (based on the GPS info of the caption) is given below:

Visible	Name	Type(s)	Distance
No	Luleå V	sublocality level 1 sublocality political	13626m
No	Äldreboende Ingridshem	establishment	290m
No	Snickare Anders Viklund i Luleå AB	establishment	225m
No	Öhemsvgen	bus station transit station establishment	225m

[4] https://en.wikipedia.org/w/api.php?action=query&list=geosearch&gsradius=500&gscoord=65.6452|22.0274&format=xml

As a result, nearby places can be used (or proposed to the user) in order to link a photo with a diary in the case that a name of a public place (restaurant, bus stop, hotel) is included in the diary text.

In a similar way to textual documents, images can be contextualized based on their content as well as their associated metadata. In this example, face detection would highlight the four people in the photo and could be used to prompt the user to identify the people; if the link to the diary entry had already been formed then the names of the people associated with it could even be suggested. Also, face clustering can detect similar (already named) faces in existing archived images and suggest them to the user. As all the personal world knowledge needed for contextualizing the photo was already gathered and stored for the diary entry the AIP containing the previously preserved personal world knowledge can be referenced to avoid duplication of information within the preservation system.

2.2 Technical Discussion

While the example discussed in the previous section helps to highlight many of the issues around contextualization, it should be remembered that the use cases within the ForgetIT project cover a much wider and varied range of tasks involving both personal and organizational preservation [Maus et al., 2013, Maus and Schwarz, 2014, Dobberkau et al., 2014]. Fortunately, regardless of the use case scenario the issues discussed in the previous example actually cover most situations we expect to encounter. Contextualization essentially boils down to the following sequence of actions:

- Process the document to extract *contextual hooks*. For all document types metadata will act as a contextual hook, providing temporal context, authorship information, location information etc. For textual documents any sequence of characters could act as a hook, although usually these will be limited to sequences annotated using techniques such as named entity recognition or term extraction, while images and video may be subject to face detection and clustering, near duplicate detection, object recognition or scene classification.

- An attempt will then be made to link each hook to the user's existing personal world knowledge as in general we assume that most references will not be to *famous* entities. For example, people mentioned in documents are more likely to be friends, relatives, or colleagues than they are to be movie stars.
  - if a link is found then it is added to the context for this document and processing moves to the next hook.

- An attempt will next be made to link the hook to general world knowledge to help place the document into a wider context.
  - if a link is found then it is added to the context for this document and processing moves to the next hook.

- If the hook has not been linked to either the personal or general sources of world knowledge then we assume that it is personal knowledge that we have not encountered before.
  - the user is prompted to provide information about the hook
  - the new information is added to the personal world knowledge and a copy is preserved
  - the link between the hook and the new information is added to the context for this document

The rest of this document focuses on the implementation of numerous components that address these steps for both text and images and general and personal world knowledge.

3 Text Contextualization

As discussed in Section 2 an important part of contextualization, of both text and images, is the process of linking a 'document' to real world entities: people, organization, location, events, etc. In the scope of text, such contextualization can be performed in various documents with different lengths, short documents like tweets or long documents like news articles. The contextualized information can be then used immediately to enhance searching within active data. In this section, we describe contextualization approaches for such short and long documents.

3.1 Contextualization for Microblog Topics Using Wikipedia Temporal Information

3.1.1 Problem Statement

Trending topics in microblogs such as Twitter are valuable resources to understand social aspects of real-world events. To enable deep analyses of such trends, semantic annotation or contextualization is an effective approach; yet the problem of annotating microblog trending topics is largely unexplored by the research community. In this work, we tackle the problem of mapping trending Twitter topics to entities from Wikipedia. We propose a novel model that complements traditional text-based approaches by rewarding entities that exhibit a high temporal correlation with topics during their burst time period. By exploiting temporal information from the Wikipedia edit history and page view logs, we have improved the annotation performance by 17-28%, as compared to the competitive baselines.

With the proliferation of microblogging and its wide influence on how information is shared and digested, the studying of microblog sites has gained interest in recent NLP research. Several approaches have been proposed to enable a deep understanding of information on Twitter. An emerging approach is to use semantic annotation techniques, for instance by mapping Twitter information snippets to canonical entities in a knowledge base or to Wikipedia [Meij et al., 2012, Guo et al., 2013], or by revisiting NLP tasks in the Twitter domain [Owoputi et al., 2013, Ritter et al., 2011]. Much of the existing work focuses on annotating a single Twitter message (tweet). However, information in Twitter is rarely digested in isolation, but rather in a collective manner, with the adoption of special mechanisms such as hashtags. When put together, the unprecedented adoption of a hashtag in a large number of tweets within a short time period can lead to bursts that often reflect trending social attention. Understanding the meaning of trending hashtags offers a valuable opportunity for various applications and studies, such as viral marketing, social behaviour analysis, recommendation, etc. Unfortunately, the task of hashtag annotation has been largely unexplored so far.

In this work, we study the problem of annotating trending hashtags on Twitter by entities derived from Wikipedia. Instead of establishing a static semantic connection between

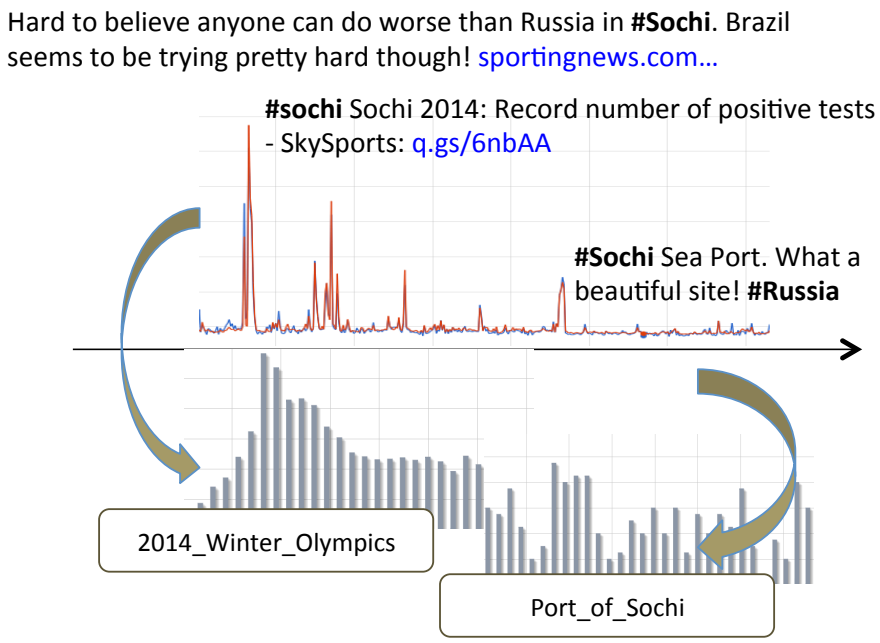

Figure 3.1: Example of trending hashtag annotation. During the *2014 Winter Olympics*, the hashtag '#sochi' had a different meaning.

hashtags and entities, we are interested in *dynamically* linking the hashtags to entities that are closest to the underlying events during trending time periods of the hashtags. For instance, while '#sochi' refers to a city in Russia, during February 2014, the hashtag was used to report the *2014 Winter Olympics* (cf. Figure 3.1); therefore, it should be linked more to Wikipedia pages related to the event than to the location.

Compared to traditional domains of text (e.g., news articles), annotating hashtags by entities poses additional challenges. Hashtags' surface forms are very ad-hoc, as they are chosen not in favour of the text quality, but by the dynamics in attention of the large crowd. In addition, the rapid evolution of the semantics of hashtags (e.g., in the case of '#sochi') makes them more ambiguous. Furthermore, a hashtag can encode multiple topics at one time period. For example, in March 2014, '#oscar' refers to the *86th Academy Awards*, but at the same time also to the *Trial of Oscar Pistorius*. Sometimes, it is difficult even for humans to understand a trending hashtag without knowledge about what is happening with the entities in the real world.

3.1.2 ForgetIT Approach

In this work, we propose a novel solution to these challenges by leveraging temporal knowledge about entity dynamics derived from Wikipedia. We hypothesize that a trending hashtag indicates an increase in public attention to certain entities, and this can also be observed on Wikipedia. As in Figure 3.1, we can identify *2014 Winter Olympics* as a prominent entity for '#sochi' during February 2014, by observing the change of user attention to the entity, for instance via the page view statistics of Wikipedia articles. We exploit

both Wikipedia edits and page views for annotation. We also propose a novel learning method, inspired by the information spreading nature of social media such as Twitter, to suggest the optimal annotations without the need for human labelling.

3.1.3 Framework

Preliminaries. We refer to an *entity* (denoted by e) as any object described by a Wikipedia article (ignoring disambiguation, lists, and redirect pages). The number of times an entity's article has been requested is called the *entity view count*. The text content of the article is denoted by $C(e)$. In this work, we choose to study hashtags at the daily level, i.e., from the timestamps of tweets we only consider their creation day. A hashtag is called *trending* at a time point if the number of tweets adopting it is significantly higher than on other days. This can be measured in different ways [Lappas et al., 2009, Lehmann et al., 2012]. Each trending hashtag has one or multiple *burst time periods*, surrounding the trending time points, where the users' interest in the underlying topic remains stronger than in other periods. We denote with $T(h)$ (or T for short) one hashtag burst time period, and with $D_T(h)$ the set of tweets containing the hashtag h created during T.

Task Definition

Given a trending hashtag h and the burst time period T of h, identify the top-k most prominent entities for h. It is worth noting that not all trending hashtags are mappable to Wikipedia entities, as the coverage of topics in Wikipedia is much lower than on Twitter. This is also a limitation of systems relying on Wikipedia such as entity disambiguation, which can only disambiguate popular entities and not the ones in the long tail. In this study, we focus on the precision and the popular trending hashtags, and leave the improvement of recall as future work.

Overview

We approach the task in three steps. The first step is to identify all entity candidates by checking surface forms of the constituent tweets of the hashtag. In the second step, we compute different similarities between each candidate and the hashtag, based on different types of contexts, which are derived from either side (Wikipedia or Twitter). Finally, we learn a unified ranking function for each (hashtag, entity) pair and choose the top-k entities with the highest scores. The ranking function is learned through an unsupervised model and needs no human-defined labels.

Entity Linking

The most obvious resource to identify candidate entities for a hashtag is via its tweets. We follow common approaches that use a lexicon to match each textual phrase in a tweet to a

potential entity set [Shen et al., 2013, Fang and Chang, 2014]. Our lexicon is constructed from Wikipedia page titles, hyperlink anchors, redirects, and disambiguation pages, which are mapped to the corresponding entities. As for the tweet phrases, we extract all n-grams ($n \leq 5$) from the input tweets within T. We apply the longest-match heuristic [Meij et al., 2012]: We start with the longest n-grams and stop as soon as the entity set is found, otherwise we continue with the constituent, smaller n-grams.

Candidate Set Expansion While lexicon-based linking works well for single tweets, applying it on the hashtag level has subtle implications. Processing a huge amount of text, especially during a hashtag trending time period, incurs expensive computational costs. Therefore, to guarantee a good recall in this step while still maintaining a feasible computation, we apply entity linking only on a random sample of the complete tweet set. Then, for each candidate entity e, we include all entities whose Wikipedia article is linked with the article of e by an outgoing or incoming link.

Measuring Entity–Hashtag Similarities

To rank the entity by prominence, we measure the similarity between each candidate entity and the hashtag. We evaluate three types of similarities:

Mention Similarity This measure relies on the explicit mentions of entities in tweets. It assumes that entities directly linked from more prominent anchors are more relevant to the hashtag. It is estimated using both statistics from Wikipedia and tweet phrases, and turns out to be surprisingly effective in practice.

Context Similarity For entities that are not directly linked to mentions (the mention similarity is zero) we exploit external resources instead. Their prominence is perceived by users via external sources, such as web pages linked from tweets, or entities' home pages. By exploiting the content of entities from these external sources, we can complement the explicit similarity metrics based on mentions.

Temporal Similarity The prior two metrics rely on the textual representation and are degraded by the linguistic difference between the two platforms. To overcome this drawback, we incorporate the temporal dynamics of hashtags and entities, which serve as a proxy to the change of user interests towards the underlying topics [Ciglan and Nørvåg, 2010]. We employ the correlation between the times series of hashtag adoption and the entity view as the third similarity measure.

Ranking Entity Prominence

While each similarity measure captures one evidence of the entity prominence, we need to unify all scores to obtain a global ranking function. In this work, we propose to combine the individual similarities using a linear function:

$$f(e,h) = \alpha f_m(e,h) + \beta f_c(e,h) + \gamma f_t(e,h) \tag{3.1}$$

Total Tweets	500,551,041
Trending Hashtags	2,444
Test Hashtags	30
Test Tweets	352,394
Distinct Mentions	145,941
Test (Entity, Hashtag) pairs	6,965
Candidates per Hashtag (avg.)	50
Extended Candidates (avg.)	182

Table 3.1: Statistics of the dataset.

where α, β, γ are model weights and f_m, f_c, f_t are the similarity measures based on mentions, context, and temporal information, respectively, between the entity e and the hashtag h. We further constrain that $\alpha + \beta + \gamma = 1$, so that the ranking scores of entities are normalized between 0 and 1, and that our learning algorithm is more tractable. The algorithm, which automatically learns the parameters without the need of human-labelled data.

3.1.4 Experiments

Dataset

There is no standard benchmark for our problem, since available datasets on microblog annotation (such as the Microposts challenge [Basave et al., 2014]) often skip global information, such we cannot infer the social statistics of hashtags. Therefore, we created our own dataset. We used the Twitter API to collect from the public stream a sample of $500,551,041$ tweets from January to April 2014. We removed hashtags that were adopted by less than 500 users, having no letters, or having characters repeated more than 4 times (e.g., '#oooommgg'). We identified trending hashtags by computing the daily time series of hashtag tweet counts, and removing those of which the time series' variance score is less than 900. To identify the hashtag burst time period T, we compute the *outlier fraction* [Lehmann et al., 2012] for each hashtag h and day t: $p_t(h) = \frac{|n_t - n_b|}{\max(n_b, n_{\min})}$, where n_t is the number of tweets containing h, n_b is the median value of n_t over all points in a 2-month time window centred on t, and $n_{\min} = 10$ is the threshold to filter low activity hashtags. The hashtag is skipped if its highest outlier fraction score is less than 15. Finally, we define the *burst time period* of a trending hashtag as the time window of size w, centred at day t_0 with the highest $p_{t_0}(h)$.

For the Wikipedia datasets we process the dump from 3rd May 2014, so as to cover all events in the Twitter dataset. We have developed Hedera [Tran and Nguyen, 2014], a scalable tool for processing the Wikipedia revision history dataset based on the Map-Reduce paradigm. In addition, we download the Wikipedia page view dataset that stores how many times a Wikipedia article was requested on an hourly level. We process the dataset for the four months of our study and use Hedera to accumulate all view counts of redirects to the actual articles.

Sampling

From the trending hashtags, we sample 30 distinct hashtags for evaluation. Since our study focuses on trending hashtags that are mappable to entities in Wikipedia, the sampling must cover a sufficient number of "popular" topics that are reflected in Wikipedia, and at the same time rare topics in the long tail. To do this, we apply several heuristics in the sampling. First, we only consider hashtags where the lexicon-based linking (Section 3.1.3) results in at least 20 different entities. Second, we randomly choose hashtags to cover different types of topics (long-running events, breaking events, endogenous hashtags). Instead of inspecting all hashtags in our corpus, we follow [Lehmann et al., 2012] and calculate the fraction of tweets published before, during and after the peak. The hashtags are then clustered in this 3-dimensional vector space. Each cluster suggests a group of hashtags with a distinct semantics [Lehmann et al., 2012]. We then pick up hashtags randomly from each cluster, resulting in 200 hashtags in total. From this rough sample, three inspectors carefully checked the tweets and chose 30 hashtags where the meanings and hashtag types were certain to the knowledge of the inspectors.

Parameter Settings

We initialize the similarity weights to $\frac{1}{3}$, the damping factor to $\tau = 0.85$, and the weight for the language model to $\lambda = 0.9$. The learning rate μ is empirically fixed to $\mu = 0.003$.

Baseline

We compare IPL with other entity annotation methods. Our first group of baselines includes entity linking systems in domains of general text, Wikiminer [Milne and Witten, 2008a], and short text, Tagme [Ferragina and Scaiella, 2012]. For each method, we use the default parameter settings, apply them for the individual tweets, and take the average of the annotation confidence scores as the prominence ranking function. The second group of baselines includes systems specifically designed for microblogs. For the content-based methods, we compare against [Meij et al., 2012], which uses a supervised method to rank entities with respect to tweets. We train the model using the same training data as in the original paper. For the graph-based method, we compare against KAURI [Shen et al., 2013], a method which uses user interest propagation to optimize the entity linking scores. To tune the parameters, we pick up four hashtags from different clusters, randomly sample 50 tweets for each, and manually annotate the tweets. We also compare three variants of our method, using only local functions for entity ranking (referred to as M, C, and T for *mention*, *context*, and *time*, respectively).

Evaluation

In total, there are $6,965$ entity-hashtag pairs returned by all systems. We employ five volunteers to evaluate the pairs in the range from 0 to 2, where 0 means the entity is noisy or obviously unrelated, 2 means the entity is strongly tied to the topic of the hashtag, and 1

	Tagme	Wikiminer	Meij	Kauri	M	C	T	IPL
P@5	0.284	0.253	0.500	0.305	0.453	0.263	0.474	**0.642**
P@15	0.253	0.147	**0.670**	0.319	0.312	0.245	0.378	0.495
MAP	0.148	0.096	0.375	0.162	0.211	0.140	0.291	**0.439**

Table 3.2: Experimental results on the sampled trending hashtags.

means that although the entity and hashtag might share some common contexts, they are not involved in a direct relationship (for instance, the entity is a too general concept such as *Ice hockey*. The annotators were advised to use search engines, the Twitter search box or Wikipedia archives whenever applicable to get more background on the stories. Inter-annotator agreement under Fleiss score is 0.625.

Results and Discussion

Table 3.2 shows the performance comparison of the methods using the standard metrics for a ranking system (precision at 5 and 15 and MAP at 15). In general, all baselines perform worse than reported in the literature, confirming the higher complexity of the hashtag annotation task as compared to traditional tasks. Interestingly enough, using our local similarities already produces better results than Tagme and Wikiminer. The local model f_m significantly outperforms both the baselines in all metrics. Combining the similarities improves the performance even more significantly.[5] Compared to the baselines, IPL improves the performance by 17-28%. The time similarity achieves the highest result compared to other content-based mention and context similarities. This supports our assumption that lexical matching is not always the best strategy to link entities in tweets. The time series-based metric incurs lower cost than others, yet it produces a considerably good performance. Context similarity based on Wikipedia edits does not yield much improvement. This can be explained in two ways. First, information in Wikipedia is largely biased to popular entities, it fails to capture many entities in the long tail. Second, language models are dependent on direct word representations, which are different between Twitter and Wikipedia. This is another advantage of non-content measures such as f_t.

For the second group of baselines (Kauri and Meij), we also observe the reduction in precision, especially for Kauri. This is because the method relies on the coherence of user interests within a group of tweets to be able to perform well, which does not hold in the context of hashtags. One astonishing result is that Meij performs better than IPL in terms of P@15. However, it performs worse in terms of MAP and P@5, suggesting that most of the correctly identified entities are ranked lower in the list. This is reasonable, as Meij attempts to optimize (with human supervision effort) the semantic agreement between entities and information found in the tweets, instead of ranking their prominence as in our work. To investigate this case further, we re-examined the hashtags and divided them by their semantics, as to whether the hashtags are spurious trends of memes inside social media (*endogenous*, e.g., "#stopasian2014"), or whether they reflect external events (*exogenous*, e.g., "#mh370"). The performance of the methods in terms of MAP scores is

[5] All significance tests are done against both Tagme and Wikiminer, with a p-value < 0.01.

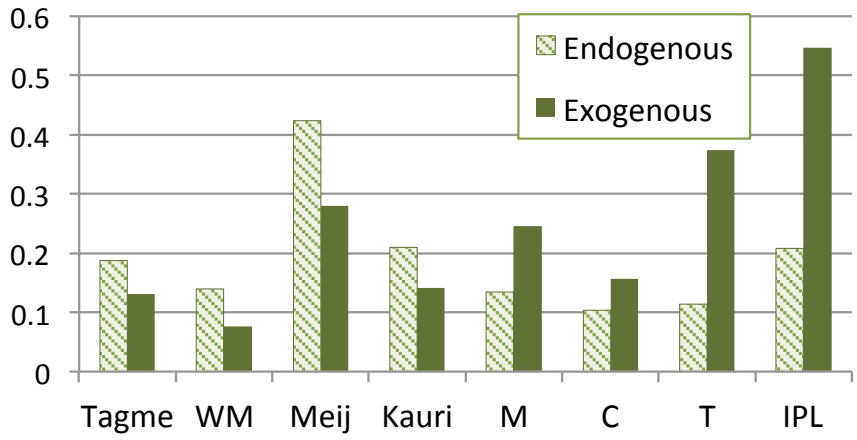

Figure 3.2: Performance of the methods for different types of trending hashtags.

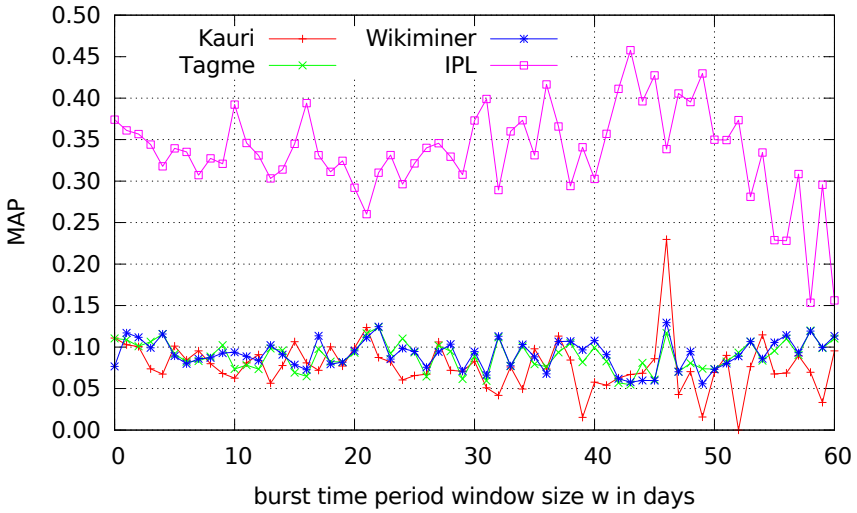

Figure 3.3: IPL compared to other baselines on different sizes of the burst time window T.

shown in Figure 3.2. It can be clearly seen that entity linking methods perform well in the endogenous group, but then deteriorate in the exogenous group. The explanation is that for endogenous hashtags, the topical consonance between tweets is very low, thus we can barely annotate further than identifying just individual concepts. In this case, topical annotation is trumped by conceptual annotation. However, whenever the hashtag evolves into a meaningful topic, a deeper annotation method will produce a significant improvement, as seen in Figure 3.2.

Finally, we study the impact of the burst time period on the annotation quality. For this, we expand the window size w and examine how different methods perform. The result is depicted in Figure 3.3. It is obvious that within the window of 2 months (where the hashtag time series is constructed and a trending time is identified), our method is stable and always outperforms the baselines by a large margin. Even when the trending hashtag has been saturated, hence introduced more noise, our method is still able to identify the prominent entities with high quality.

Contextualization

In 1954, tobacco companies ran the ad "A Frank Statement." The ad was the first in a campaign to dispute reports that smoking cigarettes could cause lung cancer and had other dangerous health effects.

Prior to 1964, many of the cigarette companies advertised their brand by falsely claiming that their product did not have serious health risks. A couple of examples would be "Play safe with Philip Morris" and "More doctors smoke Camels". Such claims were made both to increase the sales of their product and to combat the increasing public knowledge of smoking's negative health effects.

Figure 3.4: Camel advertisement (left) and contextualization information taken from Wikipedia (right).

3.2 Time-travel Translator: Automatically Contextualizing News Articles

3.2.1 Overview

Reading a current news article about a topic that you are familiar with is typically straightforward but things get worse if the article is from the past. In order to understand the article properly, acquiring context knowledge from the time of article creation is required. We call this process as *time-aware contextualization*. As an example, consider the advertisement poster from the 1950s in Figure 3.4. From today's perspective it is more than surprising that it would be actually doctors, who recommend smoking. It can, however, be understood from the context information at the right side of Figure 3.4, which has been extracted from the Wikipedia article on tobacco advertising.

Basic forms of contextualization have already been suggested in early works such as [Ferragina and Scaiella, 2010, Milne and Witten, 2008b] by adding information, which is related to the entities and concepts mentioned in the text. In our example, these techniques can produce the Wikipedia article on the mention "Camels" (camels_cigarette), "Doctors" (medical_doctors) but obtaining the contextualization information as shown in Figure 3.4 is beyond their scope. In addition, the context information should be digestible in a short time with minimal disruption from the main reading. Thus, we aim for a contextualization unit granularity which is considerably smaller than a full Wikipedia page (e.g, paragraphs).

Therefore, time-aware contextualization, which aims to associate an information item d with time-aware, coherent context information c for easing its understanding, is a challenging task. We address several properties of the context c [Tran et al., 2015]: (1) c has to be relevant for d, (2) c has to complement the information already available in d, (3) c has to consider the time of creation (or reference) of d.

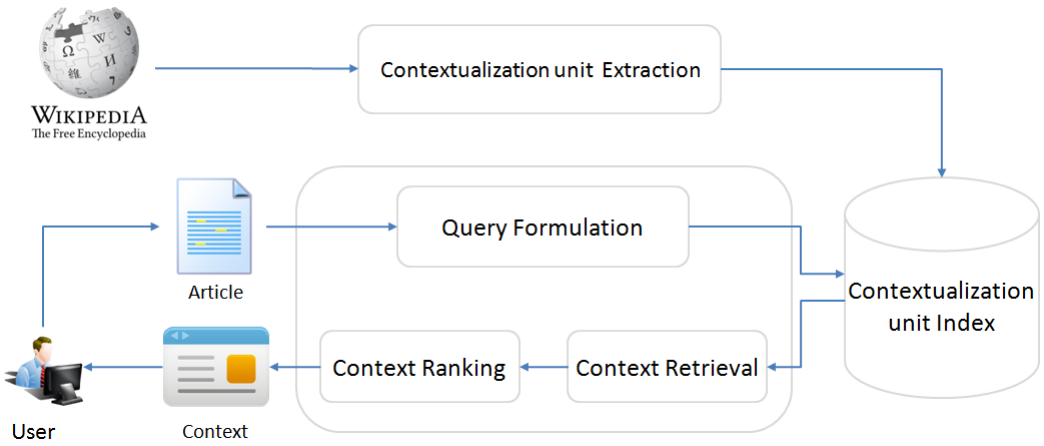

Figure 3.5: The Contextualizer Architecture.

In this work, we present a system, called Contextualizer, that provides a solution for the time-aware contextualization problem, implementing a general architecture that automatically retrieves context from user-highlighted keywords which we called *contextualization hooks* and presents them to the user in a meaningful way. To this end, we first construct appropriate queries from either the document itself or the contextualization hooks to retrieve contextualization candidates and in the next step re-rank the results by exploiting different types of information such as temporal similarity and textual complementarity. To the best of our knowledge, we are the first to present a tool that can automatically provide temporal context that is concise (not full Wiki pages) and consider the coherence of the article topic.

3.2.2 Time-aware Contextualization

The architecture of the Contextualizer system is shown in Figure 3.5. Its three main components are *context source processing* which extracts and annotates contextualization units from the context source; *query formulation* which builds the appropriate queries with using different formulation methods; and *context ranking*, which retrieves and re-ranks the results in a time-aware fashion.

Context source processing The first step is to consider which sources to be used for extracting contextualization candidates. In this demonstration, we employ Wikipedia because it is considered the largest and most up-to-date online encyclopedia covering a wide temporal range of general and specific knowledge; but the others such as News Archive can be also used. The system processes each contextualization candidate by extracting features and storing them to facilitate faster retrieval and re-ranking. We use Stanford CoreNLP[6] for tokenization, entity annotation and temporal expression extraction. In addition, *anchor* texts found in the hyperlinks are also extracted.

Query formulation The goal of the query formulation component is to generate a set

[6] http://nlp.stanford.edu/software/corenlp.shtml

of queries for a given document to retrieve contextualization candidates as input for the next component, i.e. context ranking. For building appropriate queries, we explore two families of methods, one using the document itself as a "generator", and the other using contextualization hooks as generators.

For the former family of query formulation methods, we use three strategies which exploit the document content and structure including *title* which is indicative of main topic of the document; *lead* which is the lead paragraph of the document, representing a concise summary of the document and its main actors; and *title+lead* which is a combination of the previous methods.

Based on user-highlighted keywords, i.e. contextualization hooks[7] which can be not only entity mentions, concept mentions, but also general terms and even short phrases, we consider the hook-based query formulation method by including all the hooks in a single query, representing a tailored perspective of the user's combined information needs for the document. Because the hooks are considered in the context of the document, we enrich the hook-based queries by the title of document.

Before being performed, all the queries are pre-processed by tokenization, stop-word removal and stemming.

Context ranking The queries determined given a document d in the previous step serves as a starting point for retrieving the ranked list of contextualization candidates. In order to obtain the candidates, we use query-likelihood language modelling [Ponte and Croft, 1998] to estimate the similarity of a query q with the context c.

$$P(c|q) = P(c) \prod_{w \in q} P(w|c)^{n(w,q)} \tag{3.2}$$

where w is a query term in q, $n(w,q)$ is the term frequency of w in q, and $P(w|c)$ is the probability of w estimated using Dirichlet smoothing:

$$P(w|c) = \frac{n(w,c) + \mu P(w)}{\mu + \sum_{w'} n(w',c)} \tag{3.3}$$

where μ is the smoothing parameter, $P(w)$ is the probability of each term w in the collection.

Once we have obtained a ranked list of contextualization candidates for each document, we turn to context selection where we need to decide which of the context items are most viable. Our ranking algorithm needs to balance two goals, i.e., high topical and temporal relevance as well as complementarity for providing additional information. To this end, we exploit various complementarity features and use the trained model on a set of manual labelled samples (context to document mappings) [Tran et al., 2015] to re-rank the results.

Relevance and temporal features In order to retrieve high topical and temporal relevant contextualization candidates for the document, we first consider both relevance and temporal features. For the former one, we exploit the retrieval scores of context returned by

[7] We use user-highlighted keywords and contextualization hooks interchangeably

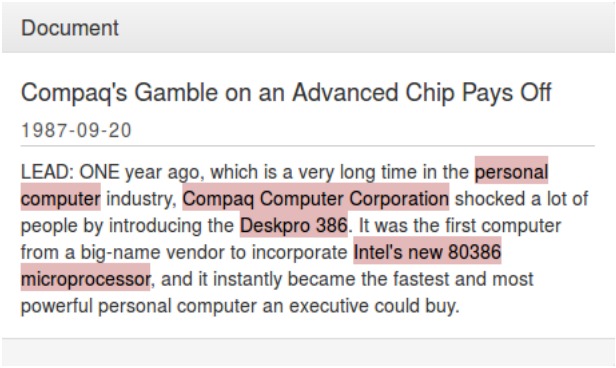

Figure 3.6: An example of user-highlighted keywords.

our retrieval model. For the later one, we apply temporal similarity measurement, i.e., TSU computed as follows

$$TSU(t_1, t_2) = \alpha^{\lambda \frac{|t_1 - t_2|}{\mu}} \tag{3.4}$$

where α and λ are constants, $0 < \alpha < 1$ and $\lambda > 0$, and μ is a unit of time distance.

Complementarity features We make use of several types of complementarity features that are confirmed an important role in contextualization including *topic diversity, text different, anchor text distance* and *geometric distance* (see [Tran et al., 2015] for computation details).

3.2.3 Demonstration of The System

Implementation Contextualizer is implemented in Java and uses Lucene for indexing and retrieving contextualization candidates. After computing all complementarity measures between each contextualization candidate and the document, we make use of the trained model on manual labelled examples using Random Forests, a learning-to-rank algorithm, to re-rank the results [Tran et al., 2015]. We implemented a web interface using the Angular[8], Bootstrap[9] frameworks and some parts from Elianto[10], all the actions are performed calling the REST API provided by the server. In order to acquire user feedback information for improving our model, we allow users to rate the candidates according to how much they are contextually related to the document in four different ratings:

- **Top relevant (3 stars)** if the candidate are topically and temporally related to the document, and complements to the main topics of the document.

- **Highly relevant (2 stars)** candidates are descriptions of the essential entities (e.g., what are entities) discussed in the document.

[8] https://angularjs.org/
[9] http://getbootstrap.com/
[10] https://github.com/dexter-entity-linking/elianto

- **Partially relevant (1 star)** candidates that provide background information about minor aspects mentioned in the document
- **Not relevant** candidates that are about different topics or repeat the same thing in the document.

In this demo, we use 51 articles that spanned a wide range of topics and publication dates from New York Times Corpus[11]. The Wikipedia dump of February 4, 2013 is used as a context source and paragraphs are considered as contextualization units. The online system are published at `http://forgetit.l3s.uni-hannover.de:9091/wikinews-webapp`.

Scenarios

In this demonstration, we will show how to retrieve additional information to the document using our Contextualizer system. As one of examples for an article of interest, suppose that a user is reading the article published in 1987 about Deskpro 386 and highlights several keywords that he/she wants more information at that time such as *"Compaq Computer Corporation"*, *"Desktop 386"*, *"Intel's new 80386 microprocessor"* as shown in Figure 3.6. In order to support the user, our system automatically suggests some candidates by using the spotting module of Dexter [Ceccarelli et al., 2013], a framework that provides all the tools needed to develop any Entity Linking technique. The user can either use these keywords or highlight his/her own ones. The Contextualizer system will use these highlighted keywords to build appropriate queries (see Section 3.2.2) such as a query *"Compaq Computer Corporation and Desktop 386 and Intel's new 80386 microprocessor"* and issue it to our retrieval component (see Section 3.2.2). First, a list of contextualization candidates are retrieved based on language modelling, and the system will decide which candidates are to be shown as context based on a variety of features and a learning to rank algorithm, i.e., Random Forests.

As shown in Figure 3.7, three contextualization candidates are returned with its Wikipedia page title and followed by the actual content. In our example, we observe that the two candidates *"When Compaq introduced the first PC based on Intel's new 80386 microprocessor, the Compaq Deskpro 386, in 1986, it marked the first CPU change to the PC platform that was not initiated by IBM..."* and *"During this period Andrew Grove dramatically redirected the company, closing much of its DRAM business and directing resources to the microprocessor business. Of perhaps greater importance was his decision to "single-source" the 386 microprocessor..."* contextually relates to the document, whereas the last one does not.

As discussed in [Tran et al., 2015], contextualization hooks can be not only entity mentions or concept mentions but also general terms and even short phrases, as shown in Figure 3.8. In this example, a user highlights two phrases *"experimental student loan program"* and *"shift the costs"*. These phrases explicitly represent the information needs of the user or, more precisely, what requires contextualization to be understood and interpreted. Here the entity linking approaches can not dealt with these contextualization hooks since

[11] `http://catalog.ldc.upenn.edu/LDC2008T19`

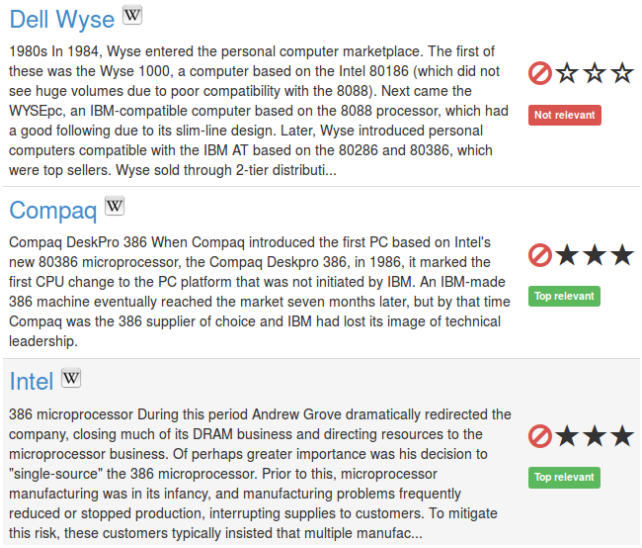

Figure 3.7: Contextualization candidates retrieved by the Contextualizer system.

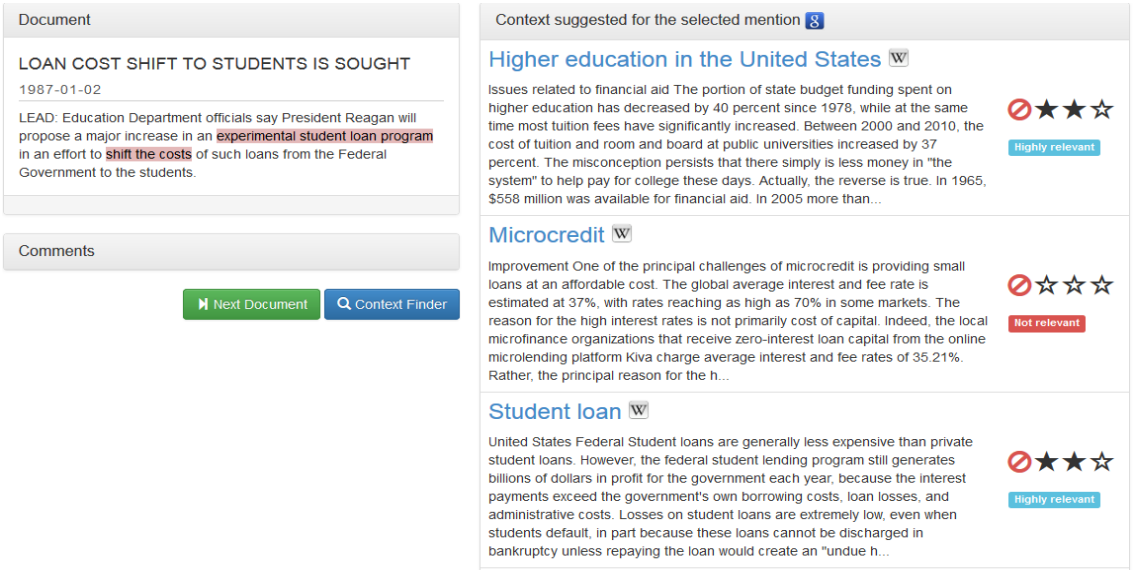

Figure 3.8: An example of the document with non-entity hooks and its contextualization candidates.

there is no explicit Wikipedia page for such phrases. In contrast to these approaches, Contextualizer system can be able to retrieve contextualization candidates based on the user-highlighted phrases as shown in Figure 3.8 (*"Higher education in the United States"* and *"Student loan"*). Again, the user can also rate the relatedness of the candidates which can be used to improve the learning process.

3.3 Text contextualization using Wikidata

The Semantic Editor, Seed enriches texts with contextual cues from world knowledge as well as personal knowledge from PIMO [Sauermann et al., 2007]. In D6.3 [Greenwood et al., 2015], we reported how Seed built on knowledge from Linked Open Data (LOD) sources to provide the previously mentioned contextual cues, which are stored as annotations in the text. On December 16th, 2014 Google announced it will retire Freebase [Bollacker et al., 2008] in mid 2015. Data from Freebase has been migrated to Wikidata [Erxleben et al., 2014]. Therefore, we decided to integrate Wikidata as an alternative LOD knowledge source in Seed.

3.3.1 What is Wikidata?

Wikidata is a linked data knowledge base, which acts as a central storage for the structured data of its Wikimedia sister projects including Wikipedia. Wikidata was created mainly to overcome the limitations in Wikipedia that appeared from the contradiction between its vision and what happens in real life [Vrandečić and Krötzsch, 2014].

There were two main reasons that made Wikidata a good candidate to be integrated as a knowledge source in Seed:

1. As mentioned earlier, Google announced on its Google+ account [12] that Freebase was being retired and its data being transferred to Wikidata. This meant the data in Freebase is no longer updated, which practically invalidates its role as a source of current world knowledge.

2. DBpedia mainly used Wikipedia's infoboxes to get its data, which potentially lead to getting inconsistent data since the information gathered could be from multiple pages for different entities. On the other hand, Wikidata is centrally used to populate Wikipedia's infoboxes, which makes Wikidata of better quality. Wikidata's data model is also richer than that of DBpedia, which makes it easy to provide additional information in a consistent way.

3.3.2 Wikidata Characteristics

Several characteristics [Vrandečić and Krötzsch, 2014] help Wikidata overcome the inconsistencies in Wikipedia. Following are some of them:

- **Open editing**: Anyone can edit the data stored.
- **Community control**: Not only data, but also its schema is community controlled.
- **Plurality**: Allows conflicting data to coexist and provides a mechanism to organize this plurality.

[12] See `https://plus.google.com/109936836907132434202/posts/bu3z2wVqcQc`

- **Secondary data**: It gathers data that is in primary sources, but with referencing these sources.

- **Multilingual data**: While Wikipedia has independent editions for each language, which results in multiple pages for the same entity, Wikidata is inherently multilingual. That means that the ID of an entity, whether it is an item (i.e. an entity) or a property is the same in any language and so extracting information about that entity is language independent and depends on its unique ID.

- **Easy access**: Wikidata's goal is to allow data to be used in Wikipedia and in external applications. So data is exposed through web services in several formats including JSON and RDF formats.

3.3.3 Wikidata's APIs

In addition to providing RDF dumps of its data, Wikidata provides many API modules for accessing it[13]. There are two important modules worth mentioning here:

- **wbsearchentities**: A module that searches for entities using labels and aliases. Parameters like the label, language, number of search results, their type(s), ...etc. are used to refine the results of queries. For example, using this API module with this query produces the following the following result shown partially:


```
{
"searchinfo": {
    "search": "cairo"
},
"search": [
    {
        "id": "Q85",
        "concepturi": "http://www.wikidata.org/entity/Q85",
        "url": "//www.wikidata.org/wiki/Q85",
        "title": "Q85",
        "pageid": 215,
        "label": "Cairo",
        "description": "capital city of Egypt",
        "match": {
            "type": "label",
            "language": "en",
            "text": "Cairo"
        }
    },
    {
        "id": "Q575306",
```


[13] https://www.wikidata.org/w/api.php

```
                "concepturi": "http://www.wikidata.org/entity/Q575306",
                "url": "//www.wikidata.org/wiki/Q575306",
                "title": "Q575306",
                "pageid": 541109,
                "label": "Cairo",
                "description": "city in Illinois, United States",
                "match": {
                    "type": "label",
                    "language": "en",
                    "text": "Cairo"
                }
            },
            ...
            ]
        }
```


- **wbgetentities**: A module to get the content all information about an entity or more using its/their IDs as parameters. Additional optional parameters like redirects, languages, property filters, ... etc. can be used. For example, using this API module with this query produces the following the following result shown partially:


```
{
"entities": {
    "Q85": {
        "pageid": 215,
        "ns": 0,
        "title": "Q85",
        "lastrevid": 295004799,
        "modified": "2016-01-23T18:39:12Z",
        "type": "item",
        "id": "Q85",
        "labels": {
            "en": {
                "language": "en",
                "value": "Cairo"
            }
        },
        "descriptions": {
            "en": {
                "language": "en",
                "value": "capital city of Egypt"
            }
        },
        "aliases": {
            "en": [
```

```
            {
                "language": "en",
                "value": "Cairo, Egypt"
            }
        ]
    },
    "claims": {
        "P1151": [
            {
                "mainsnak": {
                    "snaktype": "value",
                    "property": "P1151",
                    "datavalue": {
                        "value": {
                            "entity-type": "item",
                            "numeric-id": 16744247
                        },
                        "type": "wikibase-entityid"
                    },
                    "datatype": "wikibase-item"
                },
                "type": "statement",
                "id": "Q85$f00bc79d-4b ...",
                "rank": "normal"
            }
        ],
        ...
```


3.3.4 Implementation

We have implemented two ways for accessing the information in Wikidata, once through the APIs and another using RDF dumps loaded in a local SPARQL endpoint:

- Using Wikidata's API:
  The flow in this method of access goes as follows: First, an HTTP request is sent to the "wbsearchentities" modules using the suspected label of the entity we want to search for. This label is extracted as a named entity candidate earlier by the NLP component of Seed from the text. Then from the returned JSON response we get a list of the possible suggestions, from which we extract the unique ID of each entity which will be used in the next request, also the description and the label. Afterwards, to get further details about each item returned in the list, we issue another HTTP request to "wbgetentities" with IDs of the items returned in the previous step to get the aliases and the IDs of the type. As a last step we issue another HTTP request to

"wbgetentities" to get the labels of those types using their IDs. And then we have a list of the possible entities from Wikidata with their IDs, labels, descriptions, aliases and types. The problem with this method is issuing many requests, which often takes too long for real time interaction with the text. Due to the currently exposed functionalities of Wikidatas API, we can't formulate more complex queries to save the multiple queries using this interface. In order to perform more complex queries, the next option was implemented, which is explained in the following point.

- Using Wikidata's RDF dumps:
  Using suitable SPARQL queries, the desired entity was searched for using combinations of its "rdfs:label" and calls to the "bif:contains" function to specify patterns in the label of the entity being searched for. In order to access the "instance of" property, "rdf:type" was used and for the "subclass" property, the "rdfs:subClassOf" was used. Both of the previously mentioned properties were used to specify the type of the entity, then for the description and the alternative labels "schema:description" and "skos:altLabel" were used, respectively. The SPARQL query then, returned needed information about the possible suggestions for the target entity. With the help of Apache Jena[14], an open source Java framework for building Semantic Web and Linked Data applications. the functionality was built into the Java back-end of Seed.

Depending on the responsiveness and need for up-to-date information requirements, using the web API or the local SPARQL endpoints both have advantages and disadvantages. Using the web API provides access to the most recent version of Wikidata, but is considerably slower. On the other hand, using the local SPARQL endpoint is faster, but gets quickly outdated.

3.3.5 Conclusion

In the previous sections, we explained why we needed to integrate Wikidata as another knowledge source in Seed. We also briefly described the main characteristics of Wikidata, then explained how we implemented its integration using two access methods. Following the lines of D6.3, this contributes to world knowledge based contextualization of textual content authored in Seed.

[14] https://jena.apache.org/

4 Image Contextualization

4.1 Problem Statement

In D6.3 [Greenwood et al., 2015] we introduced an image contextualization method that enriches the image collection of a user that attended an event. That method had the following limitations:

- The user collection that is to be contextualized must be about a single event that several other users have also attended and created their own image collections.
- The other users that attended the event (called archived users in D6.3 [Greenwood et al., 2015]) must provide their collections (archived collections) to the PIMO application.
- The archived collections must have already been processed, namely they must have been temporally aligned using the time synchronization algorithm and split to subevents using the event clustering algorithm (the methods have been described in D4.3 [Solachidis et al., 2015]).

In this deliverable, we present a method that overcomes the above limitations using more elaborate feature representations and collecting the data required for the contextualization of a user collection from web resources.

4.2 ForgetIT Approach

Our updated image contextualization method is illustrated in Fig. 4.1. The user provides his textual and visual data of an event that he attended, namely 1) the image collection and 2) a textual description.

The image collection is fed into the image analysis component, which is part of WP4. Using the Caffe framework [Jia et al., 2014] and the 22-layer GoogLeNet Deep Convolutional Neural Network (DCNN) pre-trained model [Szegedy et al., 2014] we extract the *loss1/classifier* and *loss3/classifier* layers output, resulting in two $1000-$dimensional vectors. We also extract for each image its histogram in the *L*a*b** colorspace, using 8 bins for the *L** plane, 32 bins for the *a** plane and 32 bins for the *b** plane, resulting in a $72-$dimensional vector. The $2072-$dimensional concatenated feature vector of the DCNN layers output and the *L*a*b** histogram is used to represent each user collection image. We then perform Principal Component Analysis (PCA) to reduce dimensionality of this feature vector to 300.

The textual description that the user should input consists of the name of the event (e.g. Olympic games), the location (e.g. London) where the event took place, and the time (e.g. 2012) when the event took place. This information is used in order to create a set of

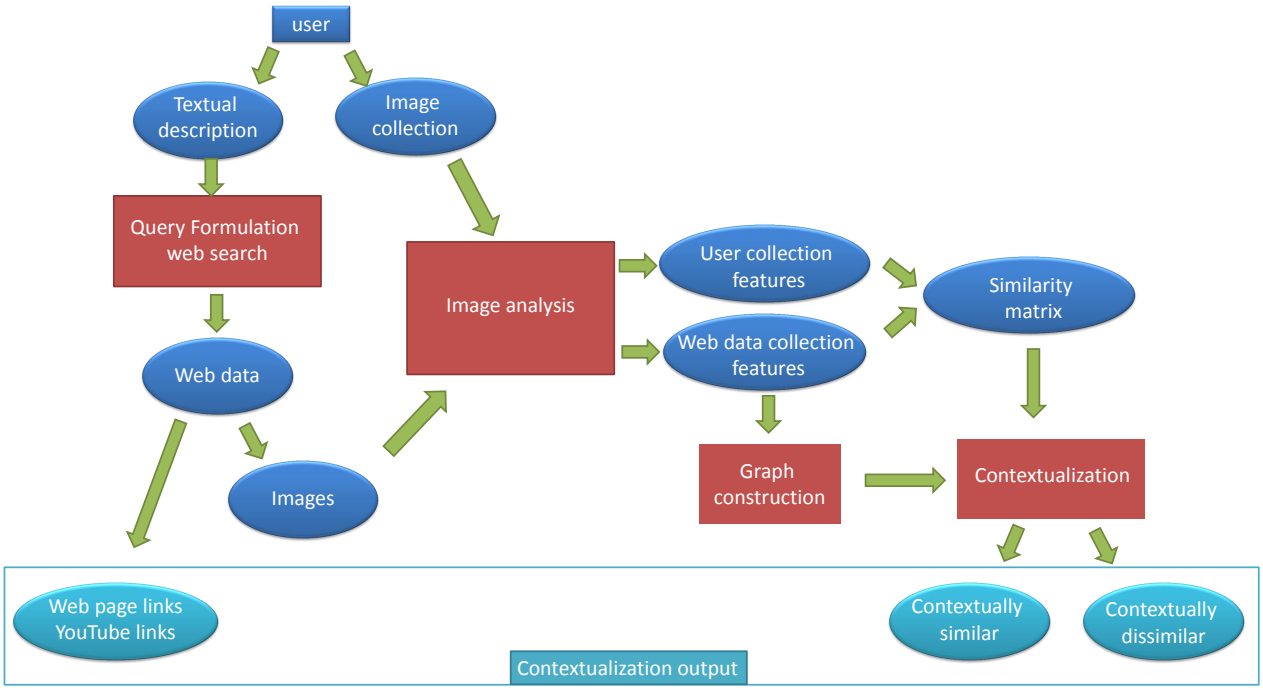

Figure 4.1: Image contextualization method workflow

queries which are sent to Web search engines and a pool of data is collected, from which the contextual information will be selected. The constructed queries are:

- A query to the Google text search engine is sent, and the P first Web pages and the Y first YouTube links that are returned are kept. Moreover, the images contained in the returned Web pages are also collected. The query to the Google text search engine is formed by combining the textual description that is provided by the user using AND (&) (i.e. event & location & time).

- A query is sent to the Bing image search engine[15], and the B top-ranked images are collected. The query is generated similarly to the previous one.

- A query is sent to the Flickr image search engine and the F top-ranked images are collected. In the case of Flickr, the Flick API allows us to add constrains to the queries. Thus, apart from the initial query (i.e event & location & time), we also send two additional queries; one limiting the location of the place where the images are taken (assuming that the images contain location coordinates) and the other limiting the time that the images have been taken (assuming that the images are assigned with the time that they were taken). In case of limiting the location the query is formulated using the event and location terms (i.e event & location) while for limiting the time only event and time terms are used (event & time).

The numbers of the returned items (P,Y,B,F) are selected empirically.

[15]Bing Image Search API is used.

With respect to the returned links (either Web pages or YouTube videos), they are directly returned to the user as contextual data and no further refinements are applied to them.

The overall set of images that is collected in the above steps is used as the pool of data that the contextualization information is retrieved from. This pool thus serves a similar purpose as the archived collections set of the method presented in D6.3 [Greenwood et al., 2015] and is simply, called *web data* from now on. The web data are fed into the image analysis component and the same features as in the case of the user-supplied image collection are extracted. We construct a similarity matrix of all images in the web data, using the Euclidean distances of their features. We consider the similarity matrix of all images in the collection as a weight matrix of a graph and find the strongly connected components of the graph. Each strongly connected component is a sub-graph in which every node is connected to every other node in the sub-graph. Therefore a strongly connected component is a group of very similar images in the web data collection.

To perform the contextualization, we construct a similarity matrix between all user images and web data images. For each image in the web data, if there is at least one image in the user-supplied image collection such that the similarity score for these two images is above t_1, then the two images are considered duplicates or near-duplicates and the corresponding web data image is not used for contextualization. If the above condition does not hold for t_1, but does hold for $t_2 < t_1$, then the corresponding image in the web data is considered *contextually similar* to the respective user-provided image or images. As stated in D6.3 [Greenwood et al., 2015], these images are similar to those of the user image collection but not identical. That information is useful since it describes the event in more details (from the user's perspective). From the rest of the web data images, if for a web data image there is a user-supplied image for which the similarity score lies in the interval (t_4, t_3) where $t_4 < t_3 < t_2$, then this web image is considered *contextually dissimilar* to the user-supplied images. As stated in D6.3 [Greenwood et al., 2015], these images are from the same event but quite different from those of the user image collection. This information is useful since it describes the event from other aspects. Finally, the remaining web data images are considered irrelevant to the user images.

It should be noted that values in the interval $(t_3, t_2]$ haven't been taken into account because these similarities values lie in the boundary between similar and dissimilar images, thus they have been omitted in order to get more accurate results since it is not clear if these images should be considered contextually similar or dissimilar with the user image collection. Also, in order not to include similar images, in the final output if several web data images belong to a single strong component of the web data graph, only one image from the same strong component is kept (the most similar to user images).

In our experiments, the values of internal parameters t_1 to t_4 have been set as follows based on experimentation: $t_1 = 0.95$, $t_2 = 0.89$, $t_3 = 0.76$ and $t_4 = 0.7$. The implemented ForgetIT method thus gets only two user-controlled input parameters a, $a > 0$, and b, $0 \geq b \geq 1$. Parameter a controls the number of the web data images that will be used for contextualizing the user collection. Parameter b specifies the percentage of the latter images that should belong to the contextually similar set; the rest of the web data images

that will be used for contextualizing the user collection will be drawn from the contextual dissimilar set.

4.2.1 Advances in Comparison to the Previous Version

The current image contextualization method extends the previous one described in deliverable D6.3 [Greenwood et al., 2015] in the following directions:

1. We replaced the local feature representations with more elaborate DCNN-based features.
2. We employed a method that overcomes the limitations identified in Section 4.1. The need for other user collections from the same event is substituted with the input of a short textual description of the event, namely the event name, the location and the time where it took place, and we use the web in order to collect the contextual data.
3. Apart from the image items that are used to contextualize the user collection, several links from related Web pages and links to video items (found in YouTube) are retrieved and provided as contextual information.

The above described method is implemented as a Rest service and is integrated in the overall ForgetIT system.

4.3 Experimental Evaluation and Comparison

We tested our method on one of the datasets used in the MediaEval SEM task [Conci et al., 2014]: the London dataset, consisting of 2142 photos capturing various subevents of the London 2012 Olympic Games. We randomly selected 129 images covering all subevents of the dataset and created the textual description as: event name “Olympic games”, location “London” and year “2012”. 588 images were downloaded from the Web and served as the web data collection.

Due to lack of ground truth data for the image collection gathered from the Web, we evaluate the contextualization results visually. A subset of the user collection is shown in Fig. 4.2(a) and a subset of the web data collection is illustrated in Fig. 4.2(b). From the web data, the images with green frame are the contextually similar images and we can notice that they are images indicating events that are contained in the user collection but from different aspects. On the other side, the images with purple frame, which are the contextually dissimilar images, show events that are not contained in the user collection. Finally the rest of the images (pink frame) are images that were not used for contextualization.

Additionally, the links of the Web pages that are returned by the web querier and point to pages which give a description of the event are:

`http://www.olympic.org/london-2012-summer-olympics`
`https://en.wikipedia.org/wiki/2012_Summer_Olympics`

https://en.wikipedia.org/wiki/London_Olympics
http://www.miniclip.com/games/london-2012-olympic-games/en/

More experimental results and evaluations in relation to our image contextualization experiments in ForgetIT can be found in [Apostolidis et al., 2015].

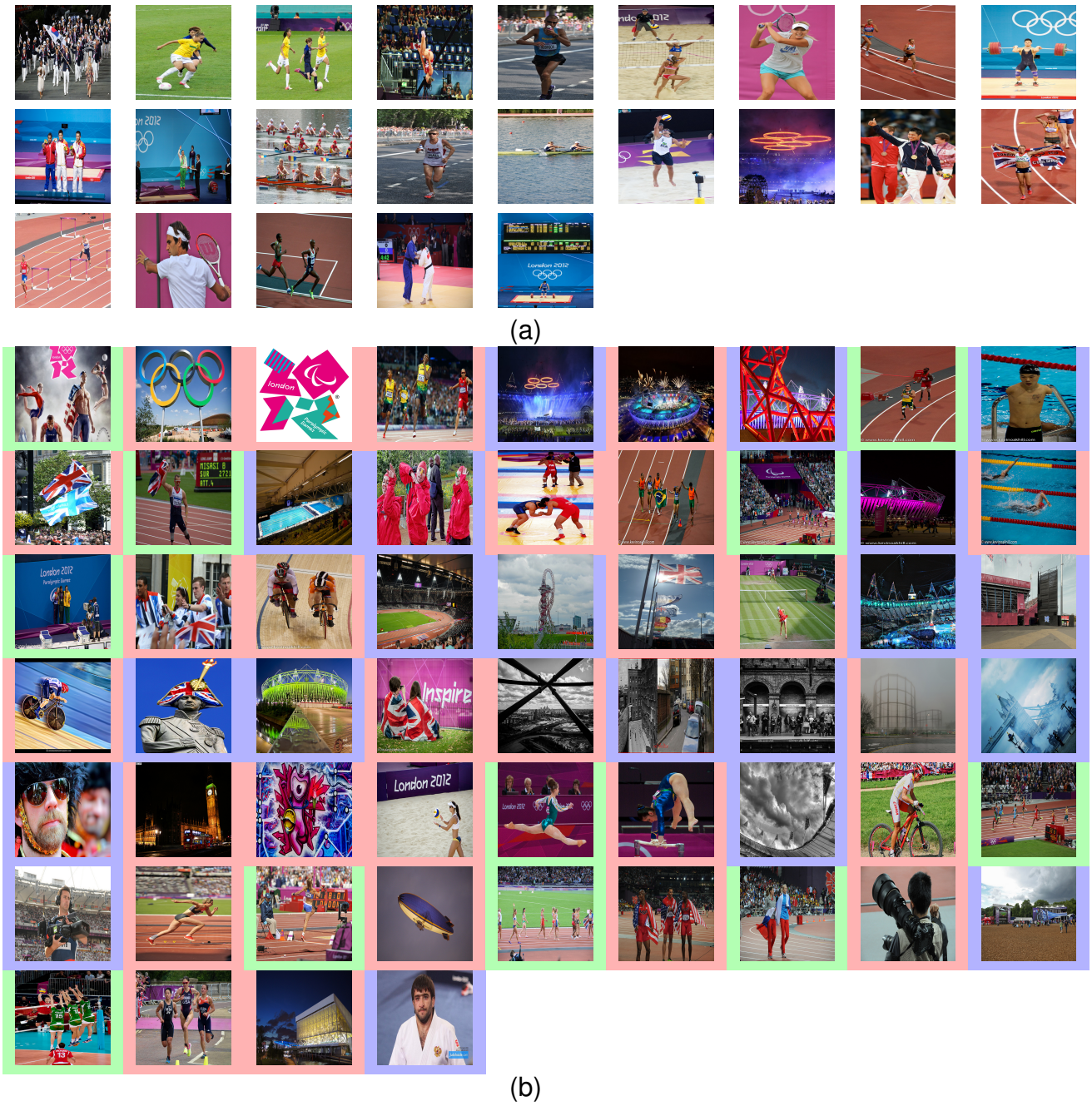

Figure 4.2: (a) User collection, (b) Web data collection - contextually similar (green), contextually dissimilar (purple), irrelevant (pink)

4.4 Software Implementation and Integration

The method is implemented as a web service hosted by CERTH. We used the Apache HTTP Server[16] as web server software and JAVA programming language for building the service. Only GET method is available.

Input description, functionalities and output results.

The image contextualization method takes as input an image collection that has been captured by a user attending an event, and its textual description. It then downloads images, Web page links and YouTube video links that are related to the event that the user attended. The purpose of this method is to enrich the initial user collection with more information about the event, information that the user might have missed when attending the event or information that describes the event in more detail. Specifically, the user provides a URL of the file containing the images of his collection. Moreover, he (or she) has to provide the name, the location and the time of the event. The method output is an Extensible Markup Language (XML) file containing the URL paths of the Web pages, the YouTube videos and the images that are selected to contextualize the collection. As far as the selected images are concerned, the URL path of a file containing the selected images is specified in the generated XML, in order for the user to decide if he/she wants to download them.

Technical Details - Instructions of Use.

The base URL of the REST service is:
`http://multimedia.iti.gr:8080/contextualization/image/get?`

Required arguments:

- **userimg,** the URL path of a zipped folder containing the image collection of the user
- **event,** the name of the event (if the event is described by more than one words they should be separated by '-')
- **location,** the city/village where the event took place
- **year,** the year when the event took place

Optional arguments:

- **country,** the country where the event took place

Error messages:

- if a required argument is not provided or misspelled: *A required argument is wrong - Please provide: userimg, event, location, year*

[16] http://www.apache.org/

- if the user image collection cannot be downloaded: *The URL path of the seed collection is wrong - PLEASE provide a zip file (only zip is supported)*

- if the user image collection consist of very few images: *Small number of images of seed collection - Contextualization cannot be applied*

- if the number of images (web data) to contextualize the collection is less than 2 the contextualization cannot be applied: *Number of images to contextualize the collection less that 2. At least one for similar and one for dissimilar should be selected*

Examples.

Example call for London 2012 Olympic-games event:

```
http://multimedia.iti.gr:8080/contextualization/image/GET?userimg=
http://multimedia.iti.gr:8080/CERTH_BIN/TEST_IMAGES/Contextualization/
london.zip&event=Olympic-games&location=London&year=2012&country=England
```

The resulting XML output contains:

<Image_Contextualization eventID=''42351''>, a unique integer 'eventID' is assigned to each user collection

<Event_info>
<event>x</event>
<location>y</location>
<year>z</year>
</Event_info>, this tag provides the textual information describing the event (e.g., x='Olympic-games', y = 'London', z = '2012')

<Web_pages>
<pg_url>http://example.com/image</pg_url>
</Web_pages>, the URLs of the retrieved Web pages

<Youtube>
<vd_url>http://youtube.com</vd_url>
</Youtube>, the URL links of YouTube videos related to the event

<Images_context_list>, the list of all selected images to contextualize the user collection
<sim>, the contextually similar selected images
<img_fl>filename of the image</img_fl>
<img_url>URL path of the image</img_url>
</sim>
<dis>, the contextually dissimilar selected images
<img_fl>filename of the image</img_fl>
<img_url>URL path of the image</img_url>

```
</dis>
</Images_context_list>
```

`<zip_url>http://multimedia.iti.gr:8080/downloadZip/rest/fileservice/download/zip?eventID=42351</zip_url>`, this tag contains the URL call of a service which the user can use to download a zip file containing the images that are selected to contextualize the image collection.

5 Context Evolution

The idea behind context evolution is to give some sense of how the world (your own personal sphere as well as in general) changes over time. Such changes clearly affect our ability to understand previously archived documents as our current world view may well be very different to that in place at the time the document was originally produced. As discussed throughout this document we roughly divide context into two forms; structured and unstructured. While both forms of context can and will evolve over time we believe that they will do so in different ways and that we need differing approaches to modelling the evolution of the two forms.

Structured data, by its very nature, is a curated resource. This curation may be an active process where an ontology or data source is knowingly updated and maintained, or it may be more of a side effect such as adding a new contact to an e-mail client which provides structured data that could be used for contextualization. In both cases though changes in the data are handled by an existing process. This applies equally to large sources of world knowledge (such as the fairly frequent updated releases of DBpedia) as it does to personal data such as that used by PIMO. In such structured resources evolution of context information is often already handled as new data supplements, rather than replaces, existing information.

In contrast unstructured context is entirely lacking in information relating to how it might have evolved over time. For example unstructured context information may take the form of topics relevant to preserved documents. Such topics are great for giving you a sense of what a specific document might be talking about but as they may not be linked to any other information exactly what they refer to and how that meaning might change from one document to another is missing. This is where extra information that can be extracted about the terms and how they evolve would be beneficial. This may involve explicitly determining the evolution of a term/topic or it might revolve more around the way the information is presented allowing users to easily infer the evolution aspects. An example will hopefully make this easier to follow.

Imagine that the data contained a simple statement along the lines of “the situation in Iran is obviously worrying and the Foreign Office is keeping a close watch on events”. It is likely that both Iran and Foreign Office would be linked to a structured source of context information (world and organizational in turn) and while useful neither gives a sense of what the problem in Iran might refer to. The date of the document will however give you some sense of what might be being referred to; for the late 1970s early 1980s it would be the revolution, since the late 1990s it is likely to refer to their nuclear research programme, and it may well refer to human rights issues regardless of the time period. In other words we need to know how the world has evolved to set the document in the correct context.

In the rest of this section we detail a number of approaches to context evolution that are able to process either structured or unstructured context.

5.1 Structured Context Evolution

5.1.1 Problem Statement

In most cases structured knowledge, that could be viewed or used as context, is produced and maintained by some form of manual process. This process may be specifically aimed at producing structured knowledge, such as the construction and maintenance of an ontology, or it could be a side effect of using an application. For example, adding the contact details for a new employee to the address book in your e-mail client has the side effect of producing structured knowledge that could be used for contextualization even though that is not the main reason for the action. Regardless of the approach the result is the same though, a structured source of knowledge that can be used for contextualization, the issue is how the data evolves over time, how we can capture this information, and how the changes can be reported or explained to the user.

5.1.2 ForgetIT Approach

Our approach to modelling the evolution of context revolves around being able to track how structured information changes over time. We take a similar approach to [Eder and Koncilia, 2004]and adopt the idea of *Valid Time*; a set of points or intervals of time in which a single fact is known to be true. Ideally this would involve having accurate timestamps associated with every piece of information but unfortunately this is not always feasible. For example, an ontology does not allow properties on relations making it impossible to timestamp the addition of a relation between two existing entities. In the very worst case scenario there may only be periodic updates of the data with only a timestamp on the entire knowledge repository. In such an extreme case the changes would need to be determined via finding all differences and each change could not be individually assigned an accurate *Valid Time*. For the purpose of further discussion we will assume that some intermediate situation holds in which the evolving data, but not necessarily all data, has a *Valid Time* associated with it.

We have previously assumed that information useful for contextualization will be stored as an ontology and so the first step in handling the evolution of this information is to specify how information on *Valid Time* should be encoded within an ontology. As previously stated relations within an ontology cannot themselves have properties so in a standard ontology it is only possible to associate timestamps with instances and not properties. Fortunately this only really affects datatype properties (i.e. those that lead to a value such as a string or number rather than an object property that leads to another instance in the ontology), and it is always possibly to re-write a datatype property as an object property if so required.

With the focus now on recording *Vital Time* for object properties it is a simple case of adding new datatype properties to each instance that record the period for which they are valid. This could use the standard `xsd:datetime` format to allow timestamps accurate to the minute (i.e. 2014-05-23T10:20:13+05:30) or could be a simpler integer representation

of a date if accuracy only to the day is required (i.e. 20140523).

For example, let us assume that we are working with an organizational ontology that includes people and the jobs they hold. Stored as triples this may give us information such as the following (using an oo prefix to stand for organizational ontology and to keep the example small):

```
<oo:Employee1> <rdfs:label> "Joe Bloggs"
<oo:Employee1> <oo:jobTitle> "Cleaner"
```

While simple this approach does not allow for changes in job title over time to be recorded. Switching from a datatype property to an object property allows us to extend the data with *Valid Time* information, stored as an integer in the form yyyyMMdd, as follows:

```
<oo:Employee1> <rdfs:label> "Joe Bloggs"

<oo:Employee1> <oo:jobTitle> <oo:Job1>
<oo:Job1> <rdfs:label> "Cleaner"
<oo:Job1> <oo:fromdate> 20080305
<oo:Job1> <oo:todate> 99991231
```

This extended ontology now details that the employee was hired as a cleaner on the 3rd of May 2008 and the far future date, in the `todate` property, shows that they are still working in that role as of the date of the ontology. Now let us assume that after just over two years Joe is promoted to a supervisory role and that his change in job title is added to the ontology.

```
<oo:Employee1> <rdfs:label> "Joe Bloggs"

<oo:Employee1> <oo:jobTitle> <oo:Job1>
<oo:Job1> <rdfs:label> "Cleaner"
<oo:Job1> <oo:fromdate> 20080305
<oo:Job1> <oo:todate> 20100607

<oo:Employee1> <oo:jobTitle> <oo:Job2>
<oo:Job2> <rdfs:label> "Cleaning Supervisor"
<oo:Job2> <oo:fromdate> 20100607
<oo:Job2> <oo:todate> 99991231
```

This approach also allows multiple overlapping pieces of information. For example, let us assume that cleaning was not a full time job and so after working for the company for a year, up until the time he was promoted to supervisor, Joe was also employed as a porter.

```
<oo:Employee1> <rdfs:label> "Joe Bloggs"

<oo:Employee1> <oo:jobTitle> <oo:Job1>
<oo:Job1> <rdfs:label> "Cleaner"
<oo:Job1> <oo:fromdate> 20080305
<oo:Job1> <oo:todate> 20100606

<oo:Employee1> <oo:jobTitle> <oo:Job2>
<oo:Job2> <rdfs:label> "Cleaning Supervisor"
<oo:Job2> <oo:fromdate> 20100607
```

```
<oo:Job2> <oo:todate> 99991231

<oo:Employee1> <oo:jobTitle> <oo:Job3>
<oo:Job3> <rdfs:label> "Porter"
<oo:Job3> <oo:fromdate> 20090305
<oo:Job3> <oo:todate> 20100606
```

Notice how each change to the ontology is now non-destructive. The only changes made are additions or setting the `todate` data property to a valid date stamp rather than the far future filler value. This means that no information is ever lost and so a single current snapshot of the ontology contains all the information related to documents written at any time during the organizations history (or at least since they adopted this approach to data storage).

What we have described so far is of course just an ontology. Having an ontology available does not automatically lead to either contextualization or to context evolution. Fortunately we have previously shown how an ontology can be used to perform contextualization. In that instance, however, the ontology did not include timestamped information so all facts were assumed to hold true at the time of use. Clearly this is no longer the case as the ontology now includes information which was true but is no longer so. For example, in the last ontology snippet given above we can see that information on Joe Bloggs being both a cleaner and porter is present in the ontology although at the current time he is only employed as a "Cleaning Supervisor". That means that we need to be careful to restrict the parts of the ontology used for contextualizing a document to only those which are valid at the time the document was created. Fortunately it is easy to add a filter to any SPARQL query to extract specific elements of an ontology. For example the following query extracts a list of employees and their job titles on the 1st of April 2009.

```
select ?e ?j where {
  ?e <oo:jobTitle> ?j .
  ?j <oo:fromdate> ?f .
  ?j <oo:todate> ?t
  FILTER (?f <= 20090401 && ?t >= 20090401)
}
```

Using our example ontology this query would return Joe Bloggs employed as a Cleaner. If we are only interested in the current status then we can utilize the far future date to select just those items that are currently valid (avoiding the filter in this case should increase performance):

```
select ?e ?j where {
  ?e <oo:jobTitle> ?j .
  ?j <oo:fromdate> ?f .
  ?j <oo:todate> "99991231"^^<http://www.w3.org/2001/XMLSchema#int>
}
```

With this ability to determine the state of the ontology at any point in time it is possible to track changes over time. For example, a time line showing the jobs held by a given person over a given time period is easy to produce. The problem is that, even in short documents, a lot of context information may be present, and if a long time has elapsed

since the document was originally written the it's context could have evolved significantly and while it may be possible to display all the changes to the user in some fashion it is likely to simply lead to information overload. The solution, although not ideal, is to manually weight domain specific properties in order to focus on showing the evolution of the most important concepts and relationships within the ontology.

5.2 Unstructured Context Evolution

5.2.1 Problem Statement

Given a query q, i.e. entity or topic, such as *Iran, Germany*, we aim at learning the context evolution of that entity during a period of time. This consists of two subtasks i.e., how to identify the context that the entity involved in a specific date, and how to determine evolution of the context over time. In this work, we consider *year* as time-unit but other granularities can be used. We discuss our solution for tackling these tasks in the following sections.

5.2.2 ForgetIT Approach

Figure 5.1 presents an overview of our approach. It consists of three main steps: *Document gathering* that collects relevant documents for the query, *Context extraction* extracts topics from those documents, and *Context Evolution* determines relevant topics in a particular time. These steps are described in more detail in the following paragraphs.

Document Gathering. The very first step is to gather relevant documents from a time-stamped collection of documents for the given query. The time-stamped corpus contains a set of articles with specific publication dates; for example New York Time articles published from 1987 to 2007, and UK Parliament debates spanning from 1935 to present. In order to obtain relevant documents, we use a query likelihood language model that determine the similarity of the query q with documents from the corpus as described in the previous section. Shortly, the likelihood of the query q from a language model estimated from a document d with the assumption that query terms are independent, is calculated as follows:

$$P(d|q) \propto P(d) \prod_{w \in q} P(w|d)^{n(w,q)}$$

where w is a query term in q, $n(w,q)$ is the term frequency of w and q, and $P(w|d)$ is the probability of w estimated using Dirichlet smoothing:

$$P(w|d) = \frac{n(w,c) + \mu P(w)}{\mu + \sum_{w'} n(w',c)}$$

where μ is the smoothing parameter, $P(w)$ is the probability of each term w in the collection.

Context Extraction. In order to extract context in which the given entity has involved, we consider topics mentioned in documents as context, and make use of Latent Dirichlet Allocation (LDA) to accomplish this task. LDA is a generative model of documents. Assume that we have a corpus D containing a set of documents, and each document d is represented by a bag of words. LDA assumes that every word in a document is generated by first picking a latent topic and then sampling from the distribution of words conditioned on the selected topic as illustrated in Figure 5.2. Probability distributions obtained after training an LDA model include $P(w|z)$, the probability word w given topic z, and $P(z|d)$, the probability of topic z given document d.

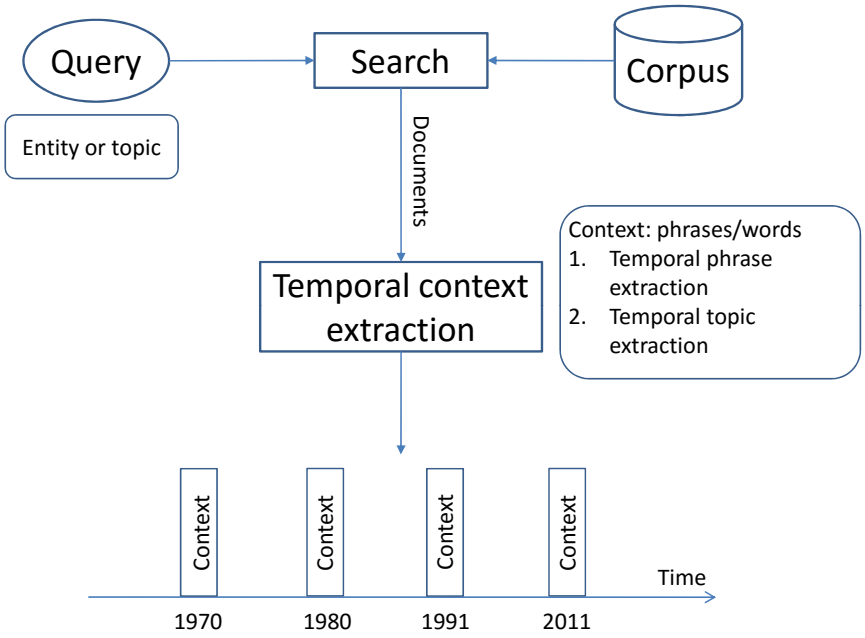

Figure 5.1: An overview of our system

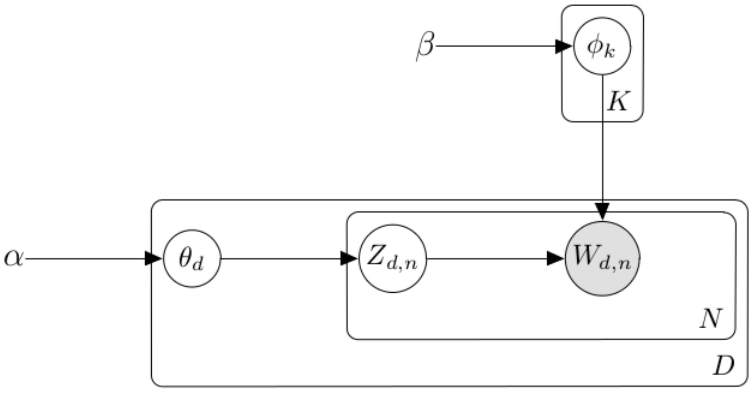

Figure 5.2: Graphical model for LDA

Context Evolution. In this step the first task is to determine the significant years for a given entity. This can be estimated in a straightforward way, by counting how many times each year has been mentioned in the documents in the time-stamped corpus. In addition, we want to see the reason why a particular year has been mentioned frequently and how the context that the entity involved in changes during the time. For this purpose, we estimate the topic distribution of each significant year. Let $P(z|d)$ be the topic distribution of each document returned by LDA, and let D_y be the set of documents mentioning y as their publication dates. We can estimate the topic distribution of each year, i.e. $P(z|y)$ using the following equation:

$$P(z|y) = \frac{1}{|D_y|} \sum_{d \in D_y} P(z|d)$$

Using the probability $P(z|y)$, we can determine a set of topics that the query entity/topics mostly involved in a specific year y. This information can be represented as context evolution timelines.

In order to determine the evolution of a specific topic over time, we can compute the probability $P(y|z)$ as follows:

$$P(y|z) = \frac{P(z|y) \times P(y)}{P(z)} \qquad\qquad P(y|z) \propto P(z|y) \times P(y)$$

where $P(z|y)$ is computed as described and $P(y)$ is estimated:

$$P(y) = \frac{\sum_{d \in D; y_d == y} 1}{|D|}$$

where $|D|$ is a total number of documents in the collection.

To make our method more clear, we consider *"Iran"* as an example query. Our system firstly identifies a list of most important years, i.e. 1991, 2010, 2014, based on how frequent the query entity was discussed in different years. For each important year, a list of related context is presented. Table 5.1 shows examples of context detected by our system.

In addition, how a detected topic was evolved or discussed over time is also presented within our system. For example, the *"iraq war saddam"* topic was mostly discussed around 1991 but we can see that it was highly mentioned again around 1999. Another example is the topic of Russia's relation with Iran. It occurred in debates around 1980 but obtained a high attention again around 2014. With unstructured context evolution information, we can see different context in different time, how they relate to the query entity and how they evolve over time. It clearly helps us in understanding the entity as shown in Figure 5.3.

Year	Topics
1991	iraq war saddam hussein house debate government statement israel east middle peace
2010	afghanistan pakistan security force iran russia sanction foreign european europe union minister
2014	syria minister government libya iran russia sanction foreign afghanistan pakistan security force

Table 5.1: Example of important years and related context (topics)

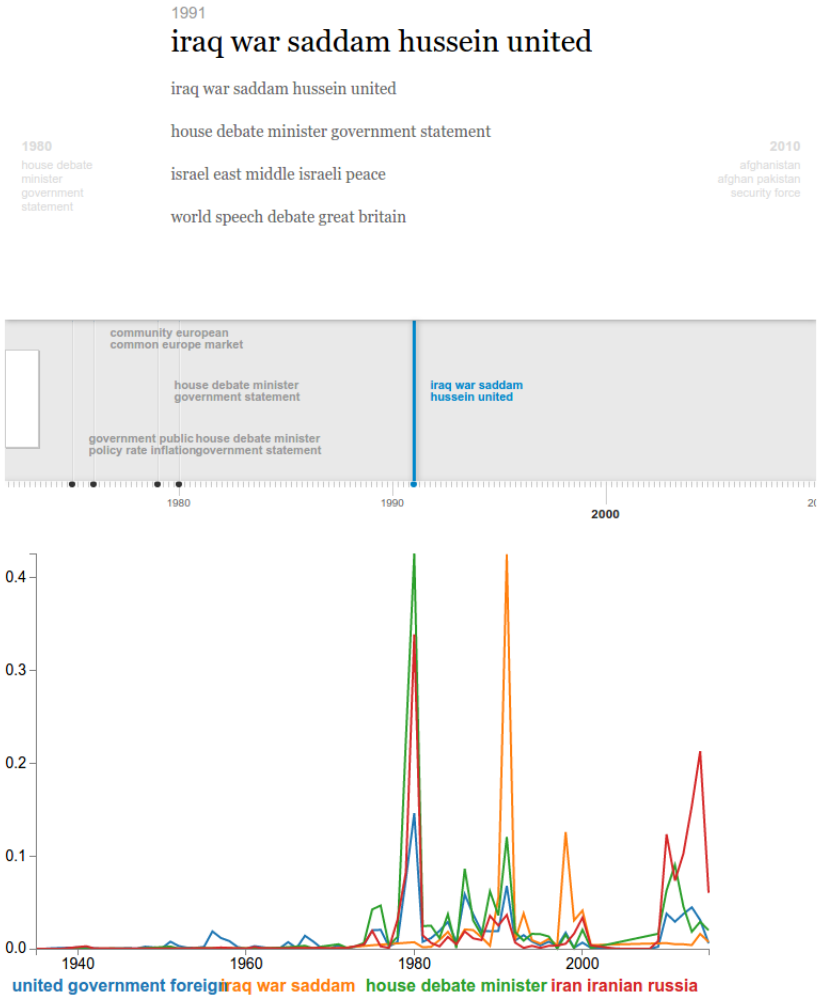

Figure 5.3: Example of contexts and their evolution for the query entity “Iran”

6 Context in Applications

6.1 Forgetful Search

Living information systems such as News archives, Web archives which are continuously augmented with fresh (new) information which show three effects: (i) they are constantly growing (ii) newer content might semantically overwrite older content (iii) and the information contained in older content ages, gradually decreasing in relevance. Search in such information collections has to take those effects into account. However, not all content ages are with the same speed.

Therefore, retrieval models are required, which use a forgetting model tailored to the respective topic. We introduce a concept of forgetful raking/search, which in addition to relevance assessment relies on a combination of different types of distance measures

- A notion of temporal distance, which reflects a general notion of information decay over time
- A notion of evolution distance, which reflects the topic specific evolution in the domain

Problem statement: Given a document collection D, and a query q, forgetful ranking returns a ranked list of relevant documents which takes into account the described effects.

Our approach In order to tackle the problem, we proposed a ranking function that combines different similarity measures between q and documents in the collection. The very first measure we use is query likelihood language model that determines the similarity of the query q with the documents, denoted by R_1. The likelihood of the query q from a language model estimated from a document d with the assumption that query terms are independent is calculated as follows:

$$R_1 = P(d|q) \propto P(d) \prod_{w \in q} P(w|d)^{n(w,q)}$$

where w is a query term in q, $n(w,q)$ is the term frequency of w and q, and $P(w|d)$ is the probability of w estimated using Dirichlet smoothing:

$$P(w|d) = \frac{n(w,c) + \mu P(w)}{\mu + \sum_{w'} n(w',c)}$$

where μ is the smoothing parameter, $P(w)$ is the probability of each term w in the collection.

The second measure is a temporal distance which reflects a notion of information decay over time, denoted by R_2. This is based on an observation that users usually want to get fresh or updated information, the newer documents are the higher the rank they obtain. Therefore, we propose to use a decay function using the publication date of a document as follows:

$$R_2 = P_t(d) = \alpha^{\lambda \frac{|t_q - t_d|}{\mu}}$$

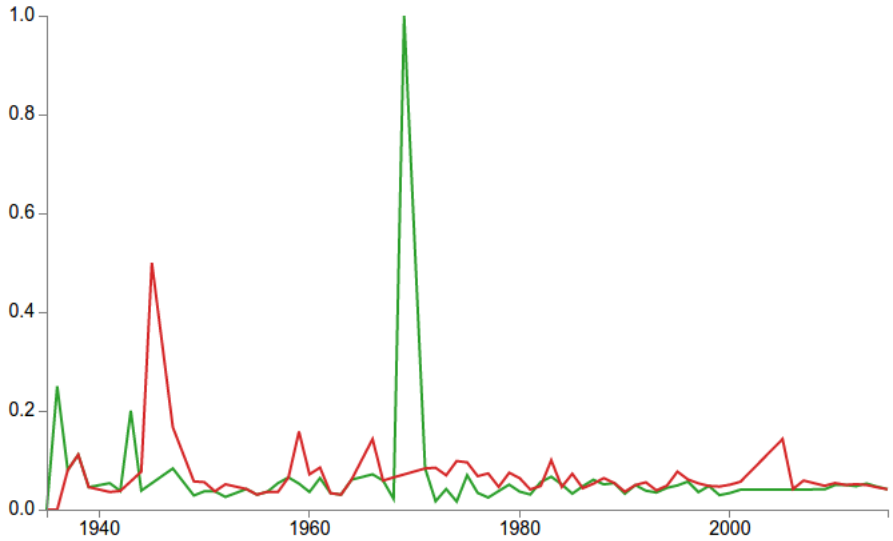

Figure 6.1: An example of topic evolution in a period of time

where t_q is the time when the query is created, t_d is the document's publication date, α and λ are constants, $0 < \alpha < 1$ and $\lambda > 0$ and μ is a unit of time distance.

The last measure we use is an evolution distance, which reflects the topic specific evolution, denoted by R_3. To make it clear, let us consider the evolution of two topics in Figure 6.1 In a period of time, i.e., from 1935 to 2015, topics are usually discussed, but in some years they are mentioned more frequently (1945 for the red topic, and 1969 for the green topic). We believe the documents that discussed the topics which are close to the peaking date, are more relevant or important than others. Hence, we propose to use a decay function using the publication date of a document and the peaking dates of the topics discussed in the document. The decay function is calculated similar to the one used in temporal distance.

$$P_e(d) = \alpha^{\lambda \frac{|t_d - p(d)|}{\mu}}$$

where $p(d)$ is the peaking date of the main topic discussed in the document d, t_d is the document's publication date and other parameters α, λ, μ are the same as in the temporal distance. In order to calculate the peaking time of a topic, we utilized the burst detection algorithm [Kleinberg, 2002] over the frequency of topics in the temporal collection. In this work, we used UK Parliament debate collection, spanning from 1935 to present (see next section for more details).

Finally, a ranking function is a combination of the three similarity measures.

$$R(d, q) = \beta * R_1 + \gamma * R_2 + (1 - \beta - \gamma) * R_3$$

where β, γ ($0 <= \beta, \gamma <= 1$ and $0 <= \beta + \gamma <= 1$) are weighting parameters to balance the importance of different similarity measures, and is set empirically in the application.

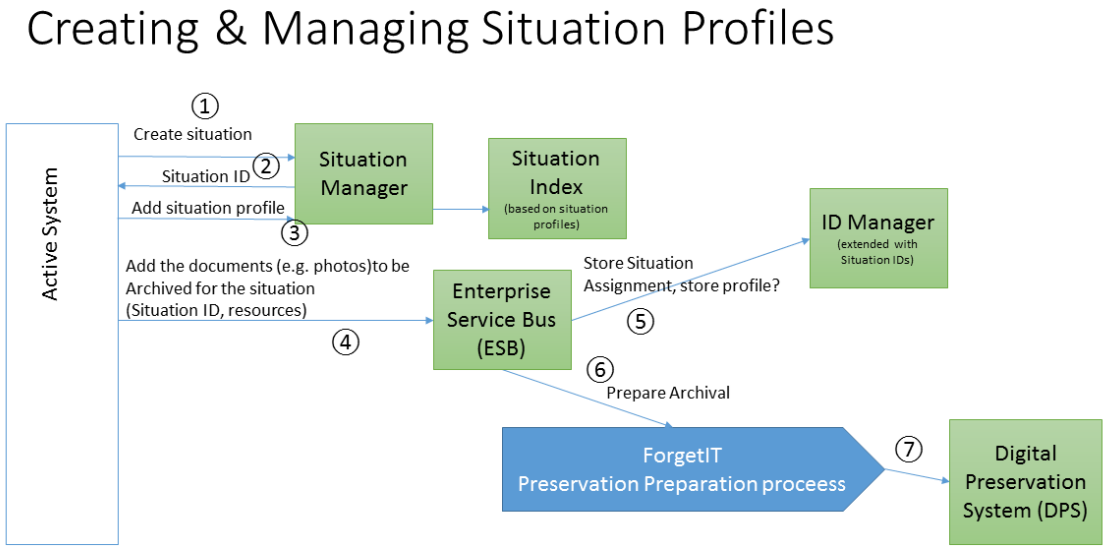

Figure 6.2: Creating and managing situations

6.2 Situation Search

Within our system collections of data (e.g. photo collections) can be associated with situations. A situation represents an event, a life situation or an experience. Examples for a situation would be a holiday trip, a birthday party or even a several year long time span like for example the time as an employee at a specific company. Situations can be hierarchical and overlapping. Each situation is represented by a situation profile which gives context information for the situation and its associated collections in the form of attributes. The association of data collections to a specific context helps users organizing their content and retrieving it with the help of situation search, a search system designed for finding situations and their associated content within the ForgetIT archive. Typically the situation profile contains context attributes like the type of the situation (holiday, family event, business trip...), the time information and the locations, persons and further entities involved. There are also attributes for describing the situation and personal memories.

Figure 6.2 depicts the sequential steps for creating a new situation within the system and associating it with content from the archive. After adding the situation profile to the situation, attributes and collections can be added or modified. Each situation receives an ID and will be stored in the middleware as well as in the archive so that a recovery is possible.

Figure 6.3 depicts the process of retrieving a situation and its associated content by using the situation search. In the first step the user uses the situation search interface to forward a search query to the system. The system returns a ranking of the situation candidates that fit the query. The user can then select one of the candidates and the system uses the

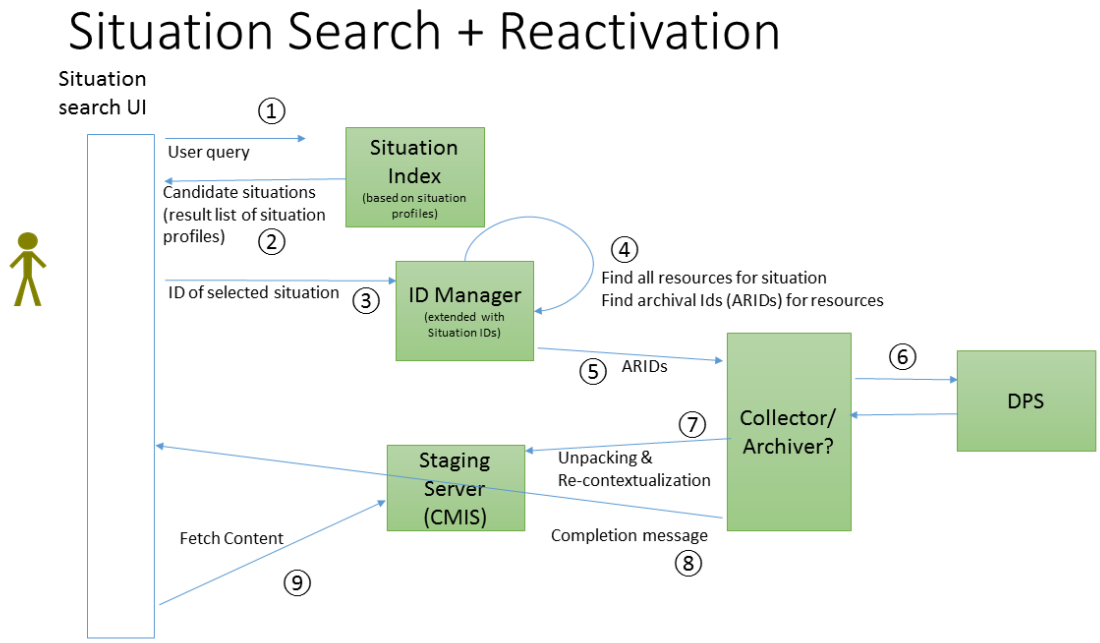

Figure 6.3: Situation search and reactivation

situation's Id to retrieve the associated content from the archive.

One of the main use cases of situation search is restoring lost data for a situation from the archive. Therefore the user only needs partial information about the situation he or she is searching for. They can use this information to search for the specific situation and reactivate the content from the archive.

The situation search is, to some extent, implemented in the ForgetIT middleware, where we store situations and their associated profiles within a Solr index. The Solr index allows the retrieval of situations using only partial information. This is especially helpful when the user has forgotten most attributes concerning the situation. For example they might have forgotten the occasion of a party but still remember the place and some of the people they met there.

We offer a restful API that provides access to the Situation Search from outside the middleware. The API comprises methods for adding, modifying and retrieving situations.

6.3 Context Aware Search

While the main aim of contextualization is to provide a context in which previously archived documents can be interpreted this does not prohibit the information from being utilized to provide an immediate benefit to those adopting contextualization within their applications and frameworks. One place where context can play an important role is within a search system by allowing related information to be used to answer search queries. In fact such

an approach often allows relevant documents to be retrieved and highly ranked which would never be found by a conventional text search engine. In the rest of this section we describe one such search system, GATE Mímir, which can make use of context information to provide a rich search experience.

GATE Mímir[17] [Cunningham et al., 2011] is a multi-paradigm information management index and repository which can be used to index and search over text, annotations, and context information. It allows queries that arbitrarily mix full-text, structural, linguistic and context queries, and that can scale to gigabytes of text. To support such queries a GATE Mímir index supports combinations of the following data types:

Text: Support for full text search represents the most basic indexing functionality, and it is required in most (if not all) cases. In GATE Mímir this is implemented as an MG4J[18] inverted index.

Annotations: The first step in abstracting away from plain text document content is the production of annotations. Annotations in this case are linguistic metadata associated with text snippets in the documents. For example, if the documents are annotated with occurrences of person and organization entities, then searches such as {Person} ", CEO of" {Organization} become possible (where {Person} and {Organization} can represent any person and organisation's name in the text respectively, and ", CEO of" is a textual string). So for example, the query would return a document containing the string "Ian Cheshire, CEO of Kingfisher plc". Exact string matching is not necessarily required: we can also use wildcards, standard binary operators, and different linguistic forms of words (e.g. root:say will match any form of the verb "say" such as "saying", "says", "said" etc.).

Context Data: As discussed previously in this and other ForgetIT deliverables we assume that information useful for performing contextualization, both world and local knowledge, stored in an ontology. Such data also allows the system to perform generalisations that are natural to humans, such as knowing that Vienna is a valid answer to queries relating to a city in Austria or Europe. While a GATE Mímir index can contain KB data, it is often more useful to link an index with one or more external repositories via public SPARQL endpoints.

It is worth noting that in this example the related data (organizational or personal depending on the application) is actually used both during the annotation of the documents and at search time, although the two uses differ substantially. The data is used to both contextualize the documents (i.e. to annotate them with instance data) and then again at search time where related information can be utilized to fulfil the search query. The main advantage of this approach over existing text search engines is that there are interesting questions which can only be answered using the additional data. The combination of text, annotations and context data allow, for example, users of the system to find such documents via queries such as:

[17] Open source and available via http://gate.ac.uk/family/mimir.html
[18] http://mg4j.dsi.unimi.it/

```
{Person sparql = "SELECT ?inst WHERE
   {?inst :birthPlace <http://dbpedia.org/resource/Sheffield>}"
} [0..4] root:say}}
```

This query basically says: *find all the places at which a Person annotation, whose URI (stored in the inst feature) is mentioned in DBpedia as being born in Sheffield, occurs within at most five words of the verb to say*. It is important to note that few if any of the documents which would match this query mention place of birth; that information would be present in the ontology used for contextualization. This nicely highlights the power of combining annotations, free text, and context data when searching a document collection.

Whilst we have shown just a single complex query that includes context hopefully this has demonstrated how powerful combining information in this way can be. This approach allows us to provide an immediate benefit to any user adopting the ForgetIT approach to preservation rather than the focus being purely on how the data can be used in some far off future to look back at old data, a situation that might, initially at least, seem less interesting to potential adopters.

6.4 Context Evolution

The context evolution might be addressed and used in other problems and applications. In this work, we present one example of them by addressing the problem of entity relatedness in different context called *Dynamic context-aware entity relatedness*.

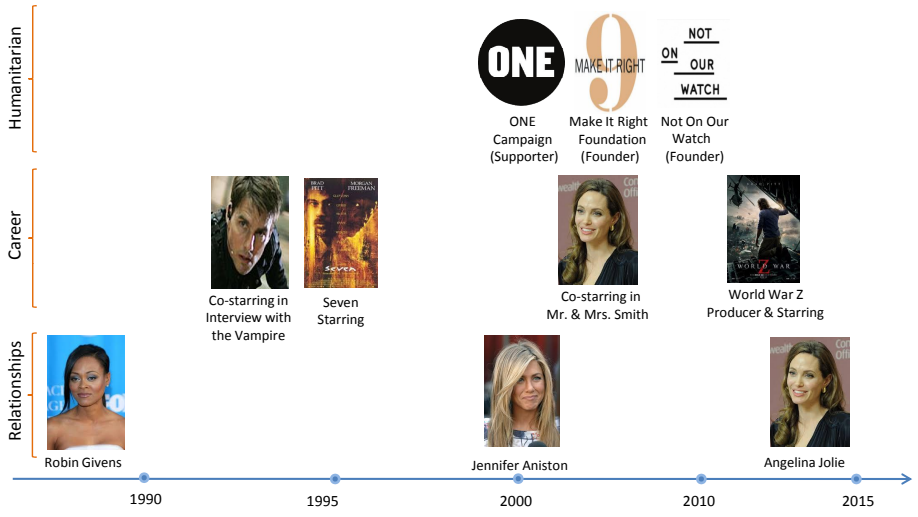

Figure 6.4: Related entities with Brad Pitt in different time periods and context

6.4.1 Introduction

The manifold relationships between entities heavily contribute to defining those entities as well as their evolution. Together, entities and their relationships form the texture, which can help in disambiguation tasks [Hoffart et al., 2011, Guo and Barbosa, 2014], in the contextualization of search results as well as in better understanding content [Tran et al., 2015].

Relationships between entities are, however, not static. While some relationships are rather robust, such as the relationship between a country and its cities, others change frequently, typically triggered by events. Yet, time is not sufficient to adequately structure the entity relationship texture. We believe that structuring entity relationships according to context - with time as just one of its dimensions - gives even better insights. This leads to the notion of context-aware entity relatedness, as it is illustrated for the entity "Brad Pitt" in Figure 6.4. While time is sufficient to structure the realm of his private relationships, there are other groups of related entities with overlapping timelines, such as the people he co-acted with in films, which relate to other contexts of his life. Such more fine granular, contextual understanding of the entity relationship texture, can be used to refine methods, which exploit knowledge about entity relationships such as entity disambiguation and the use of entity graphs for search result contextualization.

For capturing context, in our work, we follow a bottom up approach, which builds upon some forms of *micro-context* extracted from the actual surrounding of entity mentions. In this way we avoid the strict binding of entity relationship to a fixed and structured model of possible contexts. In contrast to a stronger context modelling, our approach more flexibly adapts to the context model of the user (or of the application) using our contextual entity graph: the description of relevant context given by the user is dynamically mapped to available micro-context information in the graph.

In more detail, we construct a novel contextual entity graph from Wikipedia using advanced link extraction, temporal extraction and context enrichment. In particular, each entity in the graph is attached a statistical number counting how many times it is requested, whereas each edge between two entities retains all contexts in which they are linked. This graph captures the various micro-contexts of entity relationships, together representing the evolving and context-specific relationships. In this work, we describe the problem of dynamic entity relatedness, and construction of contextual entity graph from Wikipedia.

6.4.2 Models and definitions

Entity. We are interested in entities derived from Wikipedia. Each entity has a unique identifier and its own Wikipedia article (e.g., "Brad Pitt") and all redirects from different title variants (e.g., "Pitt"). Besides the textual context of the article, each entity has information about how many times it is viewed per day since it has created by aggregating page view counts of its actual page and redirects. Formally, for each entity e, we represent e with its original Wikipedia article d as well as page view information which is a time series of view

Humanitarian and political causes	Personal life.Relationships
In May 2007, Pitt and Jolie donated $1 million to three organizations in Chad and Sudan dedicated to those affected by the crisis in the Darfur region	Pitt and Jolie announced their engagement in April 2012 after seven years together
Co-starring	**In the media**
The film stars Brad Pitt and Angelina Jolie as a bored upper-middle class married couple surprised to learn that they are both assassins hired by competing agencies to kill each other	After confirming that Jolie was pregnant in early 2006, the significant media hype surrounding the couple reached what Reuters, in a story titled "The Brangelina fever", called "the point of insanity"

Table 6.1: *Brad Pitt* and *Angelina Jolie* in different context

counts.

Entity relatedness. The relatedness quantifies the strength of the relationship (if any) between two entities, denoted as a 3-tuple $r = \langle e_1, e_2, s \rangle$, where s is a score indicating the strength. All relationships are generally considered to be persistent and time-independent such as a Spouse_Of, Capital_Of and most recent studies have been conducted in order to estimate the strengths of these static relationships.

However, the world is dynamic. The relationships between entities usually change over time by the fact entities can bind together in different time periods. For example, Brad Pitt and Jennifer Aniston are more related in 2000-2005 while Brad Pitt and Angelina Jolie are more connected from 2014 to present. Therefore, the relationships between entities need to be viewed from a more fine-grained point, by taking the time aspect into account. The intuition behind the notion of *temporal connection* is that given a time period, two entities are more likely to be temporally related if they appear close together in a large number of documents in the given period. Here, a time period T is explicitly expressed with an interval $[s_T, l_T]$.

Dynamic entity relatedness. Given two entities e_1 and e_2, we say e_1 and e_2 are dynamically related denoted as a 4-tuple $r = \langle e_1, e_2, t, s \rangle$, where t is a time window, and s is a score indicating the strength of the relationship, if they are temporally connected during the time window t.

Yet, time is not sufficient to adequately quantify entity relationship due to the fact that an entity can involve in several contexts - with time as just one of its dimensions - such as "family relationship", "career relationship" (see Table 6.1), contextual understanding of the entity relationship can be used to refined the methods which exploit knowledge about entity relationships such as entity disambiguation, entity-centric contextualization. Context can be expressed in several forms, in this paper we define a context as a 2-tuple (T, C), where T is a time window and C is a textual description of the context. For example, (C = *"World War Z is a 2013 American apocalyptic action horror film directed by Marc Forster"*, T = [2013-06-01, 2013-06-22]) is considered as a context which "Brad Pitt" participated in.

Context-aware entity relatedness. Given two entities e_1 and e_2, they are contextually related, denoted as a 5-tuple $r = \langle e_1, e_2, t, c, s \rangle$, where t is a time window, c is a text representation of context, and s is the strength of the relationship, if they both appeared in the same context c in the time period t.

We illustrate the above concepts using the following example: *We consider the real-world event "Boston marathon bombing" on April 15, 2013. Prior to this event, typical associations for "Boston" were "Chicago" or some other cities. However, during the event, "Boston" likely co-occurs in the news or social media with "Boston marathon" and "Boston marathon bombing" or other entities which involved in the event. As a result, it is contextually connected to those entities during that time period.*

Problem statement. We now formalize the dynamic context-aware entity relatedness problem. Given an input entity E in a particular context represented by a time window T, and a textual description C, discover a set of related entities and rank them according to their associations with the input entity E in the scope of context (T, C).

6.4.3 Wikipedia as a Contextual Graph

In this section, we describe how to build a contextual graph from Wikipedia which can be used in many tasks such as recommending related entity, explaining relationships between two entities. The graph was built by processing the Wikipedia dump (a snapshot from March 4, 2015). In this work, we consider Wikipedia articles as nodes (entities or concepts) in our graph. Totally, the graph had approximately 4.8 millions nodes after removing all redirect, disambiguation, category, template articles. Each node is represented by the title of the article (e.g., "Brad Pitt"), the titles of all redirect pages linking to that article (e.g., "Pitt") and all anchor texts associated with hyperlinks to the article within Wikipedia (e.g., "Brad"). Each node is considered as an entity, thus we will use the terms "node" and "entity" interchangeably. In this work, all nodes are enriched with more information by utilizing entity page view counting how many times the articles are requested on a daily level. The temporal information can be useful in many applications such as computing temporal relatedness of entities, i.e., if the page view patterns of two entities are similar in a period of time, the entities tend to be correlated during that time period.

Links (edges) between entities can be determined in several ways. Four types of links might be considered: hyperlinks and links computed from similarity of content, of category and of template [Yazdani and Popescu-Belis, 2013]. In this work, we focus on the first type, i.e., hyperlinks which are commonly used to capture topical relatedness. Every hyperlink from the content of an article towards another one indicates a certain relationship between two articles. If article P contains a link towards article Q, Q helps to understand P, and Q is considered to be related to P. In this work, we go into a further step, which attempts to analyse factors that affect the relatedness between P and Q. For each link between two entities, we enhance the link with additional information, namely, context represented by sentences in which the entities appear. In addition to contextual information, we also consider temporal expressions mentioned in the sentences. Such temporal information

will provide temporal facts about entity relationships. By considering this information, we can also answer more research questions, for example, *when* and *how* two entities are related, which have been studied recently [Fang et al., 2011, Voskarides et al., 2015].

It is important and useful to retain all hyperlinks from a source article to a target article. However, the fact is that not every sentence in an article contains explicit links to the entities it mentions, as Wikipedia guidelines only allow one link to another article in the source article's text. As an example, on the Wikipedia page for "Brad Pitt", "Jolie" occurs 32 times but only 5 of these 32 are linked to "Angelina Jolie". In order to tackle this issue, we propose an algorithm that takes a sentence as input, and iterates over n-grams that are not yet linked to an entity. If an n-gram matches a surface form of an entity, we establish a link between the n-gram and the entity. We restrict our search space to only entities that are linked from previous sentences. This way, our enriching method achieves high precision as almost no disambiguation is necessary. Finally, we obtain a contextual graph consisting of 4.8M nodes, 90M links, 101M contextual links and 112M contextual links (+10%) after enrichment.

In the contextual graph, two nodes (entities) can be linked by several contextual edges. It is noticed that in the scope of micro-context (e.g., sentences), these edges can be similar and grouped into one, because of the fact that one relationship can be expressed in different surface forms. For example, both *"Pitt has been married to actress Angelina Jolie since 2014"* and *"While Pitt stated that there was no infidelity, he also stated that he "fell in love" with Jolie on the set"* sentences express the relationship between "Brad Pitt" and "Angelina Jolie", they can be grouped into one contextual link. However, if we consider a more fine-grained context, the former sentence describes their marriage while the later one discusses the early stage of their relationship; consequently, they still can be split. Therefore, in this paper, we retain contextual links as they are mentioned in Wikipedia and leave the merging issue for future work.

7 Conclusions

7.1 Summary and Conclusions

In this deliverable the final release of ForgetIT components for contextualization are presented, based on the requirements and state-of-the-art approaches previously described [Greenwood et al., 2013] and the corresponding first and second release of components [Greenwood et al., 2014, Greenwood et al., 2015].

7.2 Assessment of Performance Indicators

In this subsection, a short explanation of how and to what extent the methods described in this deliverable fulfil the success indicators of the three expected outcomes of this workpackage, as described in the ForgetIT project Description of Work (DoW).

7.2.1 Conceptual Framework for Contextualization

The success indicators for this objective are the availability of *a framework for modelling context* and *a suitable evaluation scheme*. Over the course of the project we have presented two complementary frameworks for modelling context, the latest version of which is described in Section 2 of this deliverable, which fulfils the first of the success indicators for this objective. Developing this model did highlight that formal evaluation of any end-to-end contextualization process would be almost impossible to achieve as an acceptable result differs greatly both from person to person and for the same person over time. The only effective way to evaluate contextualization would be for a long term (decades would be ideal) user study which unfortunately falls outside the scope of this project. It is, however, possibly to evaluate the performance of each component on the links they produce between a document being archived and the local and world knowledge being used to drive contextualization and this approach has been adopted throughout the workpackage.

7.2.2 Contextualization Tools

The success indicators for this objective are the *availability of a set of context-identification modules that perform adequately* and the *successful integration of the contextualisation tools with the pilot implementations in WP9 and WP10*. The first of these indicators is fulfilled by the five different components documented and evaluated in both this and the previous deliverable [Greenwood et al., 2015]. Not every method developed within this workpackage has found a use within the pilot tools described in WP9 and W10, but the Wikidata based approach detailed in Section 3.3 of this deliverable is used within WP9 and the YODIE system documented in [Greenwood et al., 2015] is used by both WP9 [Maus et al., 2016] and WP10 [Dobberkau et al., 2016].

7.2.3 Techniques for Context Evolution

The success indicators for this objective are *the design and implementation of methods and tools to deal with information evolution*, the *degree of change in active use context that can be covered by re-contextualization*, and the *successful evaluation of information re-contextualisation when faced with information evolution*. This deliverable describes two components (see Section 5) for handling context evolution which fulfil the first of these success indicators. Neither of these described approaches places any limit on the degree of change that can be modelled. In fact both attempt to model any changes that may occur over a given time period and so fulfil the second success indicator. Unfortunately, as noted previously in this section, the use case applications have not existed long enough to build up the data that would be required for a thorough evaluation of context and certainly not long enough to include data that has evolved significantly which has made it difficult to fulfil the final success indicator. Work is, however, ongoing outside the ForgetIT project to apply the ideas of contextualization and context evolution to a pre-existing long term dataset and initial results are promising which helps to validate the approach we have taken

7.3 Lessons Learned

The research carried out within this workpackage has allowed us to learn a lot about the process of contextualization and context evolution which were not at all clear at the outset of the ForgetIT project. Chief among these lessons is the idea that context, and it's evolution, is personal in that different people require different context and over time a single individual's context needs may change significantly. For example, a banker would need very different context information when reading an article on an earthquake than a geologist. In the same way a person probably does not need to be told the name of a friend in a photo taken yesterday, but they might if the photo was taken 50 years ago. Whilst this may be a useful insight it does, unfortunately, complicate the task considerably. Such personal results require an extra level, above the extraction and evolution of context developed within ForgetIT, which takes into account both the application domain and the user. Developing personalised approaches would make for an interesting follow on research project.

7.4 Vision for the Future

The research in this workpackage has highlighted a number of interesting topics, some for which we have presented solutions in this deliverable, and others that are as yet unsolved and worthy of further investigation. A number of partners within the project are already continuing to use and further develop some of the approaches to contextualization reported in this deliverable which it is hoped will lead to further insights into this interesting area of research.

8 References

[Apostolidis et al., 2015] Apostolidis, K., Solachidis, V., Papadopoulou, O., and Mezaris, V. (2015). Photo collection contextualization. In *Multimedia Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–6.

[Basave et al., 2014] Basave, A. E. C., Rizzo, G., Varga, A., Rowe, M., Stankovic, M., and Dadzie, A. (2014). Making sense of microposts (#microposts2014) named entity extraction & linking challenge. In *4th Workshop on Making Sense of Microposts*.

[Bollacker et al., 2008] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. *SIGMOD 08 Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

[Ceccarelli et al., 2013] Ceccarelli, D., Lucchese, C., Orlando, S., Perego, R., and Trani, S. (2013). Dexter: An open source framework for entity linking. In *ESAIR '13*.

[Ciglan and Nørvåg, 2010] Ciglan, M. and Nørvåg, K. (2010). WikiPop: personalized event detection system based on Wikipedia page view statistics. In *CIKM*, pages 1931–1932.

[Conci et al., 2014] Conci, N., Natale, F. D., and Mezaris, V. (2014). Synchronization of multi-user event media (sem) at mediaeval 2014: Task description, datasets, and evaluation. In *Proc. MediaEval Workshop*.

[Cunningham et al., 2011] Cunningham, H., Tablan, V., Roberts, I., Greenwood, M. A., and Aswani, N. (2011). Information Extraction and Semantic Annotation for Multi-Paradigm Information Management. In Lupu, M., Mayer, K., Tait, J., and Trippe, A. J., editors, *Current Challenges in Patent Information Retrieval*, volume 29 of *The Information Retrieval Series*, pages 307–327. Springer Berlin Heidelberg.

[Dobberkau et al., 2014] Dobberkau, O., Damhuis, A., Doan, P., Dorzbacher, M., Goslar, J., Krasteva, V., Niederée, C., Schaffstein, S., Sprenger, S., and Wolters, M. (2014). ForgetIT Deliverable D10.1: Organizational Preservation: Application Mockups and Prototypes.

[Dobberkau et al., 2016] Dobberkau, O., Goslar, J., Gsedl, I., and Wolters, M. (2016). ForgetIT Deliverable D10.4: Organizational Preservation Report.

[Eder and Koncilia, 2004] Eder, J. and Koncilia, C. (2004). Modelling Changes in Ontologies. In *Proceedings of the On The Move - Federated Conferences (OTM 2004)*.

[Erxleben et al., 2014] Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., and Vrande, D. (2014). The Semantic Web ISWC 2014. volume 8796.

[Fang et al., 2011] Fang, L., Sarma, A. D., Yu, C., and Bohannon, P. (2011). Rex: Explaining relationships between entity pairs. *Proceedings of the VLDB Endowment*.

[Fang and Chang, 2014] Fang, Y. and Chang, M.-W. (2014). Entity linking on microblogs with spatial and temporal signals. *Trans. of the Assoc. for Comp. Linguistics*, 2:259–272.

[Ferragina and Scaiella, 2010] Ferragina, P. and Scaiella, U. (2010). Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM '10*.

[Ferragina and Scaiella, 2012] Ferragina, P. and Scaiella, U. (2012). Fast and accurate annotation of short texts with Wikipedia pages. *IEEE Softw.*, 29(1):70–75.

[Goetz, 2011] Goetz, T. (2011). Harnessing the Power of Feedback Loops. *WIRED Magazine*.

[Greenwood et al., 2013] Greenwood, M. A., Ceroni, A., Kanhabua, N., Mezaris, V., Niederée, C., Nilsson, J., and Papadopoulou, O. (2013). ForgetIT Deliverable D6.1: State of the Art and Approach for Contextualization.

[Greenwood et al., 2014] Greenwood, M. A., Ceroni, A., Mezaris, V., Niederée, C., Papadopoulou, O., and Solachidis, V. (2014). ForgetIT Deliverable D6.2: Contextualization Tools - First Release.

[Greenwood et al., 2015] Greenwood, M. A., Ceroni, A., Petrak, J., Gorrell, G., Mezaris, V., Papadopoulou, O., Solachidis, V., Eldesouky, B., and Maus, H. (2015). ForgetIT Deliverable D6.3: Contextualisation Tools - Second Release.

[Guo et al., 2013] Guo, S., Chang, M.-W., and Kıcıman, E. (2013). To link or not to link? A study on end-to-end tweet entity linking. In *NAACL-HLT*, pages 1020–1030.

[Guo and Barbosa, 2014] Guo, Z. and Barbosa, D. (2014). Robust entity linking via random walks. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, CIKM '14.

[Hoffart et al., 2011] Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust disambiguation of named entities in text. In *EMNLP '11*.

[Jia et al., 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

[Kleinberg, 2002] Kleinberg, J. (2002). Bursty and hierarchical structure in streams. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02.

[Kuny, 1998] Kuny, T. (1998). The Digital Dark Ages? Challenges in the Preservation of Electronic Information. *International Preservation News*, (17).

[Lappas et al., 2009] Lappas, T., Arai, B., Platakis, M., Kotsakos, D., and Gunopulos, D. (2009). On burstiness-aware search for document sequences. In *KDD*, pages 477–486.

[Lehmann et al., 2012] Lehmann, J., Gonçalves, B., Ramasco, J. J., and Cattuto, C. (2012). Dynamical classes of collective attention in Twitter. In *WWW*, pages 251–260.

[Maus et al., 2013] Maus, H., Dobberkau, O., Wolters, M., and Niederée, C. (2013). ForgetIT Deliverable D9.1: Application Use Cases & Requirements Document.

[Maus and Schwarz, 2014] Maus, H. and Schwarz, S. (2014). ForgetIT Deliverable D9.2: Personal Preservation Mockups.

[Maus et al., 2014] Maus, H., Schwarz, S., Eldesouky, B., Jilek, C., Wolters, M., and Loğoğlu, B. (2014). ForgetIT Deliverable D9.3: Personal Preservation Pilot I: Concise Preserving Personal Desktop.

[Maus et al., 2016] Maus, H., Schwarz, S., Jilek, C., Wolters, M., Rhodes, S., Ceroni, A., and Gür, G. (2016). ForgetIT Deliverable D9.5: Personal Preservation Report.

[Meij et al., 2012] Meij, E., Weerkamp, W., and de Rijke, M. (2012). Adding semantics to microblog posts. In *WSDM*, pages 563–572.

[Milne and Witten, 2008a] Milne, D. and Witten, I. H. (2008a). Learning to link with Wikipedia. In *CIKM*, pages 509–518.

[Milne and Witten, 2008b] Milne, D. and Witten, I. H. (2008b). Learning to link with wikipedia. In *CIKM '08*.

[Owoputi et al., 2013] Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In *NAACL-HLT*, pages 380–390.

[Ponte and Croft, 1998] Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *SIGIR '98*.

[Ritter et al., 2011] Ritter, A., Clark, S., Etzioni, O., et al. (2011). Named entity recognition in tweets: an experimental study. In *EMNLP*, pages 1524–1534.

[Sauermann et al., 2007] Sauermann, L., van Elst, L., and Dengel, A. (2007). PIMO – A Framework for Representing Personal Information Models. In Pellegrini, T. and Schaffert, S., editors, *I-SEMANTICS Conference 5-7 September 2007, Graz, Austria*, J.UCS, pages 270–277. Know-Center, Austria.

[Schwarz et al., 2011] Schwarz, S., Marmann, F., and Maus, H. (2011). Extracting personal concepts from users' emails to initialize their personal information models. In *Knowlege-Based and Intelligent Information and Engineering Systems*, volume 6882 of *LNCS*, pages 430–439. Springer.

[Shen et al., 2013] Shen, W., Wang, J., Luo, P., and Wang, M. (2013). Linking named entities in tweets with knowledge base via user interest modeling. In *WSDM*, pages 68–76.

[Solachidis et al., 2015] Solachidis, V., Papadopoulou, O., Apostolidis, K., Ioannidou, A., Mezaris, V., Greenwood, M. A., and Maus, H. (2015). ForgetIT Deliverable D4.3: Information Analysis, Consolidation and Concentration Techniques, and Evaluation - Second Release.

[Szegedy et al., 2014] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*.

[Tran et al., 2015] Tran, N. K., Ceroni, A., Kanhabua, N., and Niederée, C. (2015). Back to the past: Supporting interpretations of forgotten stories by time-aware re-contextualization. In *WSDM '15*.

[Tran and Nguyen, 2014] Tran, T. and Nguyen, T. N. (2014). Hedera: Scalable indexing, exploring entities in Wikipedia revision history. In *ISWC*, pages 297–300.

[Voskarides et al., 2015] Voskarides, N., Meij, E., Tsagkias, M., de Rijke, M., and Weerkamp, W. (2015). Learning to explain entity relationships in knowledge graphs. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, ACL '15.

[Vrandečić and Krötzsch, 2014] Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

[Yazdani and Popescu-Belis, 2013] Yazdani, M. and Popescu-Belis, A. (2013). Computing text semantic relatedness using the contents and links of a hypertext encyclopedia. *Artificial Intelligence*.

Glossary

DCNN Deep Convolutional Neural Networks. 34

PCA Principal Component Analysis. 34

XML Extensible Markup Language. 39