www.forgetit-project.eu

ForgetIT

Concise Preservation by Combining Managed Forgetting and Contextualized Remembering

Grant Agreement No. 600826

Deliverable D6.2

Work-package	WP6: Contextualization / Decontextualization
Deliverable	D6.2: First Release of Tools for Contextualization
Deliverable Leader	Mark A. Greenwood
Quality Assessor	Olivier Dobberkau
Estimation of PM spent	15
Dissemination level	Public
Delivery date in Annex I	31st January 2014
Actual delivery date	4th February 2014
Revisions	6
Status	Final
Keywords:	context, text, images, ontologies, preservation

Disclaimer

List of Authors

Andrea Ceroni (LUH)

Mark A. Greenwood (USFD)

Vasileios Mezaris (CERTH)

Claudia Niederee (LUH)

Olga Papadopoulou (CERTH)

Vassilios Solachidis (CERTH)

Contents

Executive summary

This is the first of a series of prototype deliverables which will be produced by WP6 to document the contextualization components developed for ForgetIT. The five components described in the rest of this document (four for processing text, and one for images) are initial prototypes and as such will lay the foundations for future work, and should not be considered to be our complete solution to the problem of contextualization.

Although this deliverable's main focus is the prototype components, it also includes a thorough discussion of what the ForgetIT project views as contextualization, including a more formal model than that included in D6.1.

1 Introduction

In the previous workpackage deliverable, D6.1, we gave an overview of what we meant by contextualization in the context of ForgetIT. Further discussion and the development of the first prototype components have allowed us to refine these ideas further. This deliverable is mostly concerned with giving a brief overview of the prototype components, but this and the following section are included to document our current ideas about what constitutes contextualization and the requirements this places on the components we are developing.

The most crucial issue in formalizing the contextualization problem is that what constitutes *context*, and how it is computed, will differ not only between media types but also between approaches to contextualization which work over the same media type. This makes it challenging to define a project wide formalism or storage format that will take into account the differing approaches we intend to pursue. Regarding this issue, the role of this deliverable is twofold. Firstly, in Section 2, we propose a high-level conceptual model of the contextualization framework in an attempt to give a unifying view of contextualization over different media types and approaches, as well as to mention the different aspects that characterize the problem. This is followed, in Sections 3 and 4, by descriptions of the prototype components developed within the project which show how the general model can be applied to different instances of the contextualization problem.

Regarding actual data formats, it is of course beneficial to have some restrictions in place that govern the contextualization data we intend to generate and preserve. For the moment, our solution has been to keep the data format flexible with the caveat that, where possible, it should be stored in open, preferably human readable, formats. This allows for the context to be understood when items are retrieved from storage even if the re-contextualization components, or the ForgetIT platform as a whole, are no longer available. Clearly this becomes more important the longer items remain in the archive, although it is hoped that storlets (see WP7) may allow for the migration of contextualization data to newer formats should the need arise.

Detailed examples of the initial prototypes of such approaches to contextualization are documented throughout the rest of this deliverable, but the following is a brief summary in order to round off this overview. As already discussed different media types are likely to produce different forms of contextualization data. Algorithms which process text may, for example, generate contextual information by extracting definition like sentences from a large corpus (e.g. Wikipedia), or may link entities to a structured knowledge source (DBpedia being the obvious, but not only, candidate). The contextualization of images and videos may generate similar data (e.g. a picture of a well known building with a reference to Wikipedia or DBpedia in which the building is described), or may use other images as context (e.g. an image of an architectural detail on a large building is contextualized by a wider angle photo showing the detail in situ).

The major outstanding issue regarding this approach to contextualization is how much context is required to allow a document to be interpreted as close as possible to its intended interpretation? If we consider, just as an example, textual documents then it is right

to assume that contextualization would include providing defining information about entities or relations mentioned within the document. It is also clear that any entity definition will also contain entities or relations which could in turn be contextualized. The question then, is how far away from the original document should contextualization reach? If, for example, contextualization is performed against an ontology then this question can be rephrased as which portion of the ontology (defined by the number of edges followed) needs to be preserved. While the remainder of this deliverable does not address this issue directly, there are ongoing discussions within the project which will lead to a number of user experiments in which the amount of context required for understanding will be explored leading to data sets which can be used for testing future versions of the contextualization components.

2 General Contextualization Model

2.1 Problem Statement

The *context* can be generally defined as the information that surrounds one or more *pieces of information* and that influences its interpretation. Different contexts can lead to different *interpretations* of the same piece of information. Context can change over time, for instance as a consequence of term and concept evolution [1, 2]. A piece of information can also be subject to different interpretations depending on who creates or interacts with it, since every person is endowed with a (more or less) different *background knowledge*. In addition, the background knowledge of the same person might change over time. We explicate these facts by referring to $C\left(t\right)$ as the context existing at time t, and to $I\left(i, C\left(t\right), B\left(t, u\right)\right)$ as the interpretation of the piece of information i done within the context C and with the background knowledge B of user u at time t.

Let us assume that a piece of information has to be archived at a given time point. In order to ensure a consistent interpretation in the future, where the context is likely to be different from its original context, parts of the information constituting the original context has to be made explicit and archived with the piece of information at archiving time. This is the process of *contextualization*, which can be formally stated as follows.

> For a piece of information i, **contextualization** is the process of providing a certain amount of additional information $c^{+}(i, t_o) \subset C(t_o)$, called *context information*, such that i can be interpreted/understood by a user u_f in a future time point t_f, where a new context $C(t_f)$ exists, in a similar way as it has been interpreted in the original context $C(t_o)$:
>
> $$I(i, C(t_o), B(t_o, u_o)) \cong I((i \oplus c^{+}(i, t_o)), C(t_f), B(t_f, u_f)) \tag{2.1}$$
>
> where the function $I(i, C, B)$ refers to the interpretation of the piece of information i in context C with background knowledge B, and u_o is the creator of i. $I(i, C\left(t_o\right), B\left(t_o, u_o\right))$ can be also referred to as *intended interpretation*. Note that c^{+} is a subset of the original context.

In principle, the context information that needs to be added to i depends on both the time distance between t_f and t_o, denoted $t_f - t_o$, and a prospective user u_f. For instance, the older i becomes (i.e. $t_f - t_o$ increases), the more additional information is required in order for i to be understood. Also, at the same future time t_f, different users u_f endowed with different background knowledge might require different additional information to understand i. The main problem is that neither t_f or u_f can be known in advance.

This general formulation of the contextualization problem is difficult to use in practice, since the concepts of piece of information, context, context information, interpretation, and contextualization are very abstract. In the following sections we propose a set of preliminary simplifications and operational choices to make the above defined model usable in practice.

2.2 Content and Context Information

Still from a high level perspective, we introduce the concepts of *content information*, *descriptive context information*, and *explanatory context information* to make distinction between the information contained in i and in c^+.

> Given a piece of information i, **content information** $c(i)$ is the information contained in i. This can be information explicitly contained in i as well as information that can be extracted from i.

> Given a piece of information i, **context information** $c^+(i,t) \subset C(t)$ is the additional information associated to i at time t defined as:
>
> $$c^+(i,t) := c^+_{desc}(i,t) \oplus c^+_{expl}(i,t) \tag{2.2}$$
>
> where the *descriptive context information* c^+_{desc} is the additional information required to describe i, and the *explanatory context information* c^+_{expl} is the additional information required to explain i.

Note that the time dependency of the context information has a twofold meaning. First, since the context C itself evolves over time, the information that can be extracted from C to be part of c^+ changes over time as well. Second, the contextualization time is in general different from the creation time of a piece of information. The amount of context information to be associated to the piece of information might increase with the difference between contextualization and creation time. Simplifications and operational choices for both descriptive and explanatory context information will be proposed in the following sections.

2.2.1 Content Information

The content information of a document is defined as the information that is either contained in a document or that can be extracted from a document, such as entity mentions, depicted persons, temporal references, topics, concepts, entity relationships, and terms. It should be clear, however, that contextualization is not the same as information extraction (IE). The task may rely on tools commonly used for IE but contextualization is about providing more details about a document other than the entities, topics, and other features it may contain. For example, when processing a document IE tools may be used to find relevant entities, but contextualization must go a step further and link those entities to some other source of knowledge so that the entities can be better understood in isolation which in turn allows for a deeper understanding of documents in which those entities occur.

2.2.2 Descriptive Context Information

The descriptive context information of a document at a given time point is defined as a set of document metadata like the creation date, author, and position of the document within the collection hierarchy at that time.

While any preservation system should retain document metadata, such as the time the file was created and last modified, there are many other pieces of information about the document that can also help with defining its context. For example, the author of the document is an important part of its context, as would the name of the project, product, or holiday with which the document was associated. While such metadata may already be being automatically generated (i.e. documents created with Microsoft Office record the author name if configured correctly) as part of the file format, it is important that this information is extracted (or prompted for) and stored as context within the ForgetIT archive so that it can be accessed and searched over without requiring items to first be retrieved from the archive. We currently envisage that this form of context will be collected via the user-facing tools developed for ForgetIT as it is information that it will often not be possible to extract in any automatic fashion from the documents themselves. As such this type of context is not covered by the prototypes outlined in the rest of this deliverable although it is expected that some details will be reported in later WP6 deliverables as well as in deliverables from WP9 and WP10.

2.2.3 Explanatory Context Information

Explanatory context information is the amount of information, generally time dependent, added to a document in order to better understand it in the future. Computing this kind of information is the core of any contextualization component, since it is what makes contextualization different from simply extracting features from documents and retrieving their metadata. Indeed, the output of every prototype component is treated as explanatory context information for the input document.

Although the data model is currently different for each prototype component, each one of them provides the explanatory context information for a given document by exploiting both its content and its descriptive content information. The next section will provide a unifying operational model for the different contextualization methods proposed based on the definitions above.

2.3 Operative Contextualization Model

In this section we propose an operative model for contextualization, starting from the general concepts introduced in Sections 2.1 and 2.2. In particular, our model is centred on the explanatory context information, since it is the novel aspect that makes contextualization different from other information extraction approaches.

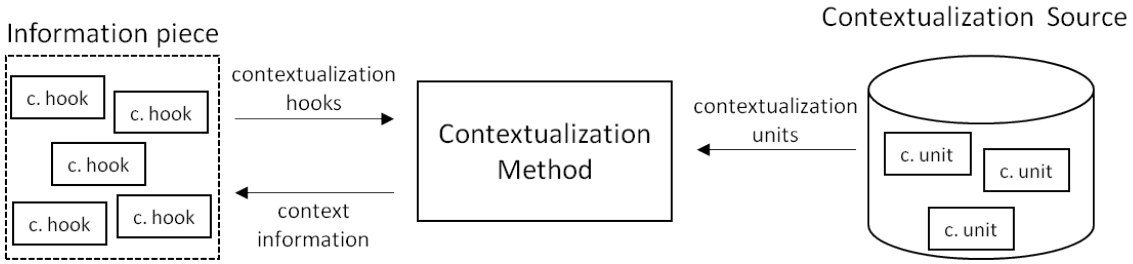

Figure 1: Core concepts of the operative contextualization model.

2.3.1 Operative Simplifications

For the moment, we simplify the general model described in Sections 2.1 and 2.2 in the following way.

- We assume that the user background knowledge does not affect contextualization
- We assume that a model for the interpretation function is not available

Given these simplifications, the problem of contextualization is restricted to the computation of explanatory context information of a piece of information i on the basis of its content information and the current context.

2.3.2 Core Concepts

In this section we describe the core concepts that underlie our operative contextualization model, whose high-level overview is depicted in Figure 1.

The starting point of the contextualization process are the items to be contextualized. Those *pieces of information* might be entire text documents (or parts of them), images, other types of documents as well as document collections. The content of those pieces of information is assumed to contain a set of elements (or they can be extracted from the content), which we call *contextualization hooks*. Those contextualization hooks provide seeds for the contextualization. For textual documents, hooks can be a variety of things including entity, event and concept mentions, temporal references, or phrases within the text, as well as implicit topics and relationships between those items (e.g. co-occurrence in the same sentence) that are extracted through information extraction tools. For images possible contextualization hooks are persons and objects depicted, locations, creation dates as well as other information that can be extracted from the images and/or from the associated metadata. Note that contextualization hooks are considered as a subset of the content information defined in Section 2.2.1.

A further core concept is the *contextualization source*. This is the source from which the context information for the contextualization process is taken. Possible contextualization sources are reference ontologies and knowledge bases such as Wikipedia, the

collection the piece of information itself belongs to, as well as external collections. In any case, we model the contextualization source as a set of *contextualization units*, which are used to add information to the given piece of information based on the contextualization hooks. What a contextualization unit is, clearly depends on the contextualization source as well as on the chosen contextualization method. If it is an ontology, then they might be subject-predicate-object facts between entities; in case of a collection of textual documents, contextualization units might be sentences or paragraphs; or they can be images within an image collection.

Given a piece of information and a contextualization source, we might be interested in answering questions like: can we know more about a given entity? What did he/she/it do at a given time? Do two entities have any relationship regarding the main topic of the document? Are there other documents that can help in understanding the given one? In order to answer these questions, we have to define a *contextualization method*. Irrespective of its possible implementations, which might be different in terms of complexity and supported data types, the input–output perspective of the contextualization method does not change: given a piece of information, its hooks, and the context source, it provides a set of contextualization units that helps in understanding and interpreting the information piece. Indeed, the output of the contextualization method represents the explanatory context information defined in Section 2.2.3.

2.3.3 Contextualization Method Definition

Based on the concepts and framework introduced above we can now define a contextualization method in terms of these concepts in an operative way.

Given a set of information pieces I, the set of all possible contextualization units CS, and the set of possible time points T, a contextualization method M can be defined as $M = (I_M, H_M, CS_M, h_M, m_M, r_M)$ where

- $I_M \subset I$ is the set of information pieces to be contextualized
- $H_M \subset c(I)$ is the set of possible hooks in I, which is a subset of the content information c(I) as defined above and depends on the type of hooks considered;
- $CS_M \subset CS$ is the set of contextualization units to be used for contextualization, i.e. the contextualization units contained in the contextualization sources used in the methods;
- an annotation function $h_M : I_M \mapsto P(H_M) \times P(H_M \times H_M) \times (T \cup \{undefined\})$, which is used to identify contextualization hooks in the piece of information under consideration as well as relationships between those hooks. The annotation also identifies a point in time;
- a matching function $m_M : H_M \times (T \cup \{undefined\}) \mapsto P(CS_M)$, which identifies contextualization candidates for the contextualization hooks identified by the annotation function; The inclusion of the value ”undefined” captures the use of matching

function, where time information is not explicitly considered, e.g. if both the hooks and the contextualization refer to the same time.

- a ranking function $r_M : CS \mapsto \Re$, which ranks the contextualization candidates identified by the mapping function with the aim to create a concise, useful and diverse contextualization;

Given this definition the contextualization problem can be framed as a combination of an annotation, a matching and a ranking problem.

Based on the above framework and the definition of the contextualization method, different contextualization methods can be described in terms of the type of contextualization hooks and sources used and in terms of the annotation, matching and ranking function defined. An example is an entity or concept-centric contextualization, where the contextualization hooks are entity or concept references (mentions) in the text. The annotation method h could, for example, be an named entity recognition method. In this case the matching method m would identity relevant information about the entities (an related entities) e.g. with DBPedia serving as the contextualization source.

To be more concrete we take a look at the contextualization methods presented in section 3 and 4 of this deliverable:

- Contextualization of text in Section 3.1: The I_M is composed from textual information, the contextualization hooks are entity mentions in the text, the contextualization source is open linked data, where the contextualization units are entity descriptions and the environment as extracted from an ontology. The matching method focuses on entity disambiguation for linking to the right entity on the ontology.

- Re-contextualization in section 3.2: The I_M are textual sources more precisely news articles from the past. The contextualization hooks are phrases in the News articles, the contextualization source is Wikipedia, which has been pre-processed in order to create an sentence level index. The contextualization units are thus indexed Wikipedia sentences equipped with temporal information. The presented method considers time both in the annotation function h as well as in the mapping function m.

- Contextualization in the personal scenario in section 3.3: In this case the items to be contextualized are items from the personal desktop. Since we are working with the semantic desktop in the ForgetIT project we are in the lucky situation that there are already given contextualization hooks defined by links from the resource to the user ontology (PIMO). The contextualization source is the PIMO ontology, i.e. parts of the ontology attached to a resource is used for contextualization. The main challenge of the functions m and r are thus to decide how much of the ontology to archive as context together with the resource.

- Event-based Contextualization: This work focuses on using events as contextualization hooks.

- Contextualization of images in section 4: In this case I_M is composed from image collections. As contextualization hook model vectors of images are used. The contextualization source are other collection of images together with their metadata. The matching function m relies on clustering methods for images identifying promising image collections for adding further image to the source collection.

The respective contextualization methods are described in more detail in the following sections.

3 Prototype Components for the Contextualization of Text

3.1 Contextualization via Disambiguation

3.1.1 Overview

The context of a text document could be viewed as knowledge about entities and events which appear with the document, but which is not explicitly stated. For example, the relationships between people may not be explicitly stated but would help a reader to fully understand the content of the document.

Our approach to this problem is to use Linked Open Data (currently DBpedia) as the source of knowledge external to a document. Rather than simply referring to entities via their public URIs we take a *relevant* subset and store this along side the text documents being archived. This ensures that the knowledge stored as context is immune to changes or deletion.

The first stage in such a process of contextualization is to uniquely identify each entity and/or event within a document with respect to a specific ontology. This process is essentially a case of disambiguation and allows us, for example, to determine if the textual mention "Paris" is referring to the capital of France[1], a city in Texas[2], or maybe even a well known socialite[3].

In our research into disambiguation [3, 4] we have focused on determining which of a given set of DBpedia entries, which share a common lexicalization, is actually being referred to. Similar to state-of-the-art methods, our algorithm uses the textual context, in which the particular candidate entity appears, to calculate a number of similarity metrics:

- *String similarity*: edit distance between the text string (such as *Paris*), and the lexicalizations of the entity URIs (e.g. *Paris* and *Paris, Texas*).
- *Structural similarity*: calculated based on the ontology and instance property values in the Linking Open Data[4] (LOD) resource. For example, the number of direct or indirect links between mentioned concepts.
- *Contextual similarity*: calculated based on the probability that two words have a similar meaning, based on random indexing [5].
- *Commonness*: number of mentions of a specific URI as anchor text in Wikipedia (based on the commonness metric for Wikipedia pages [6], also referred to as popularity [7]). This is the equivalent to assigning the most frequent sense in word sense

[1] http://en.wikipedia.org/wiki/Paris
[2] http://en.wikipedia.org/wiki/Paris,_Texas
[3] http://en.wikipedia.org/wiki/Paris_Hilton
[4] http://linkeddata.org/

disambiguation. However, unlike [7], for efficiency we do not use Google queries as additional evidence.

A combined score is derived using a weighted sum of these separate metrics. In the case of a tie, i.e. candidate URIs with the same overall score, the URI with the highest commonness score is selected and any remaining ties are broken by selecting the most specific instance according to the ontology, which in the current system is DBpedia.

This process of disambiguation generates a set of instance URIs that describe the main entities within a document, but it is the wider ontology that acts as the document context. As mentioned previously a relevant subset of DBpedia is archived as context. We currently have two fairly simplistic approaches to sub-setting that can be used to extract a documents context from DBpedia. Both approaches use the set of disambiguated URIs as their starting point and select nodes within the graph that are no more than a set number of edges from the starting nodes within the graph; current experiments follow up to 3 edges. The first algorithm does this blindly following all edges regardless of their type. The second algorithm is more selective and only follows those edges which are within the DBpedia namespace. Both algorithms select a very large section of the ontology, and clearly need to be refined further.

3.1.2 Usage Instructions

This contextualization component is currently provided as a standalone library rather than as a RESTful webservice. While the base disambiguation code (and does) run as a web service, the current approaches to sub-setting the ontology take quite a while to work; depending on the depth this will often exceed standard server timeout settings.

A single `contextualize` method takes in a piece of text and returns a `Context` object which encapsulates both the set of disambiguated entity URIs found within the text, as well as a collection of RDF triples which encode the wider context as extracted from DBpedia.

The code is available to download from `http://downloads.gate.ac.uk/forgetit/CviaDPrototype.zip`.

Functional description	Disambiguation based contextualization
Input	Text Document and DBpedia
Output	RDF Triples
Limitations/Scalability	None
Language/technologies	Java
Hardware Requirements	N/A
OS Requirements	Any OS with Java support
Other Requirements	N/A

Table 1: Component Summary

3.1.3 Future Work

Future development of this component will focus on two main areas. Firstly we currently perform disambiguation against DBpedia. While this may be sufficient in the organizational usecase where many documents may refer to well known entities, it is likely to be less useful in the personal use case. Personal documents are more likely to refer to friends and relatives, who are unlikely to occur within DBpedia. In this case performing disambiguation against the users personal ontology would be more appropriate. The second area of development will focus on the ontology sub-setting. The intention is to run a series of user experiments to determine how far, and for what types of information, from an original document a user is likely to navigate in order to fully understand the content.

3.2 Re-Contextualization without Original Context

3.2.1 Overview

This prototype component addresses the problem of document Re-Contextualization when the original context is missing, i.e. the document has not been contextualized before being archived. Addressing such a worst-case scenario is useful for, at least, two reasons. First, any future approach to Re-Contextualization, exploiting the presence of the original document context, can incrementally work on top of this component. Second, there are a lot of documents, in principle any document before and outside the ForgetIT framework, which have not been stored along with their original context.

The typical scenario covered by the component consists in understanding and properly interpreting an old document, e.g. a 20 years old news article, by providing further information, coming from an external Knowledge Base. We chose Wikipedia[5] as Knowledge Base, but the approach remains valid in principle with any other document collection. We used articles from the New York Times Annotated Corpus[6] to test our component. A deeper investigation into the problem leads to different issues and research questions. Which parts of the document the user does not understand? How these depend on user background knowledge and on time? How to exploit the information already contained in the document to drive the retrieval of additional, complementing, and different information? Does its amount depend on time and/or on user background knowledge? This component establishes the basis for answering such questions.

3.2.2 System Description

Given an input document to be re-contextualized, predicting textual elements that need further information is a non trivial task, since they might depend on the user background

[5] http://www.wikipedia.org
[6] http://catalog.ldc.upenn.edu/LDC2008T19

knowledge and on the publication date of the document (or any other temporal reference within it). Our first approach simplifies the problem by assuming that the user reads the abstract of the document, supposed to be a representative summary of the whole document, and annotates those elements requiring additional contextualizing information. To this end, a graphical interface has been developed to collect user annotations.

Each input annotation is used to query a sentence-level index of Wikipedia, where temporal expressions and entities in sentences have been annotated and can be queried through a faceted search. In order to retrieve information that refers to an annotation in the context of the article, each query can be expanded by considering entities, concepts, topics, temporal expressions, and terms belonging to the article. The first version of the component exploits the article title to expand queries, under the assumption that the title is a good representative of the main topic of the article. Topic modelling approaches will be investigated and exploited in future versions of the component.

As an example of the query formulation process, let us consider the news article ”Clinton Tells Moscow Crowd That Future Won't Be Easy” having the following abstract:

> Pres Clinton, in meetings with Pres Boris N Yeltsin and in speech to students at Moscow university, delivers sober lecture about hard realities of capitalism and price of international support for relief from Russia's current financial crisis; says there is no easy escape from financial catastrophe that has closed banks, left millions of workers unpaid and caused collapse of Government; officials say Clinton's meetings with Yeltsin were businesslike; only concrete accords to be announced at meetings are agreements to reduce both nations' plutonium stockpiles and to share data on ballistic missile launchings; both were negotiated in weeks leading up to summit meeting.

If the text “financial crisis” has been annotated for being contextualized, then the query

> text:“financial crisis” AND text:(Clinton Tells Moscow Crowd That Future Won't Be Easy)

is performed, where text is a field representing the text of the sentences in the index. Double quotes are used to require an exact matching between the original annotation and the sentences in the index. This is expected to filter out not relevant sentences that only contain part of the annotation. We do not put such constraint for the article title, since it would dramatically reduce (or probably empty) the result set.

The sentences retrieved for every query are then gathered into a unique result set and re-ranked through a ranking function that takes into account three different metrics.

- *Relevance*: it is the relevance of a retrieved sentence with respect to the query, computed by exploiting the Probabilistic Retrieval Framework presented in [8].
- *Complementarity*: according to [9], it balances similarity and difference between two sentences. For every query, complementarity is assessed between a retrieved sen-

tence and every sentence in the document that contain the annotation that triggered the query.

- *Temporal Similarity*: already exploited as ranking feature in [10], this metric takes into account the difference between one or more temporal expression in each retrieved sentence and the publication date of the document. The more the temporal expression within a sentence is temporally close to the publication date of the document, the higher is the rank of the sentence.

The open parameters of the overall ranking function, e.g. the weighting factors of the above mentioned metrics, are learned through *Learning to Rank* techniques [11].

Finally, the output of the component is the set of the top-k sentences according to the ranking criteria described above. For the moment, the value of k has to be explicitly specified by the user. The prediction of the value of k given an estimation of the user background knowledge is left as a future work.

3.2.3 Usage Instructions

Functional description	Document re-contextualization
Input	XML and JSON documents
Output	XML document
Limitations/Scalability	N/A
Language/technologies	Java, Apache Solr
Hardware Requirements	N/A
OS Requirements	Any OS with Java support
Other Requirements	N/A

Table 2: Component Summary

As already stated, the component requires two inputs: a textual document and a set of annotations. The input document is an XML file, which is assumed to contain both the text and the publication date of the document. Annotations and their positions within the text are stored in JSON format. The output of the component is an XML file: it is the original XML file representing the input document enriched with one child element for every contextualizing sentence.

Regarding the integration within the ForgetIT framework, the component can be used whenever a document passes from the archival information system to the active system. The component might also exploit the output of the Extractor component, consisting in relevant information extracted from the document, like named entities, concepts, temporal references, and important terms. Currently, since the integration has not been done yet, the component exploits external information extraction tools. By exploiting Wikipedia as Knowledge Base, it is envisioned that the component will mostly be used in an organizational scenario. Here, documents like old news articles and web pages contain globally

known people, locations, and facts, which are likely to be mentioned in Wikipedia. On the contrary, personal documents might contain references to people, facts, and locations that are known only within a personal scenario, e.g. family, friends, colleagues, and consequently they are not mentioned in Wikipedia.

3.3 Contextualization in the Personal Scenario

3.3.1 Overview

This component performs contextualization in the personal scenario, where a personal knowledge base is available. The availability of a personal knowledge base is crucial, since personal documents usually contain mentions to persons, organizations, events, and locations that are only known in a personal scenario. In order to achieve an effective contextualization, they should be registered within the exploited knowledge base. Our component exploits the PIMO ontology as knowledge base to contextualize personal documents, which are modelled as PIMO resources.

Given a resource to be contextualized, the basic form of personal contextualization identifies all the PIMO resources explicitly related to the input resource via the relations defined in PIMO. That is, the context of a resource is a set of relations, explicitly present in the knowledge base, between the given resource and other resources. In the following section the skeleton architecture of the component is described. Future and more complex prototypes, while keeping the same architecture, will both take the content of the resource into account (e.g. linking resource mentions to PIMO ontology and contextualizing them) and discover implicit resource relations. Future versions will also work in the organizational scenario by handling TYPO3 CMS data as it is done for PIMO resources. Since any contextualizing resource can be in turn contextualized, an open issue to be investigated is how much the process has to be iterated to achieve an effective and concise contextualization.

3.3.2 Architecture and Usage Instructions

The high-level architecture of the component is sketched in Figure 2. The knowledge base is derived from user data and interaction log of PIMO semantic desktop systems. The data are sent to the cache in the contextualizer via RESTful web services, and are stored in JSON format. Here the data conforms to the data model of the managed forgetting (Details in Section 2, Deliverable D3.2). Essentially speaking, “static” data such as relations between resources, resources’ properties and structure of the system are stored in RDF triples, where a fact is represented as a first-order relation between two entities (called a subject and object respectively). We call these triples RDF facts - in the Contextualizer such facts should center around a given resource of interest, and constitutes an *RDF graph* called Resource Context Graph or Resource Graph for short. For the

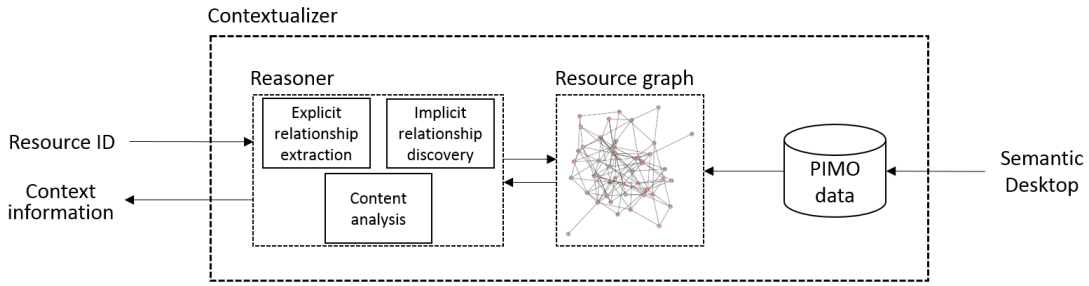

Figure 2: High-level architecture of the contextualization in the personal scenario.

Functional description	Contextualization in personal scenario
Input	Resource ID
Output	JSON file
Limitations/Scalability	N/A
Language/technologies	Java, OpenRDF Sesame, PIMO
Hardware Requirements	N/A
OS Requirements	Any OS with Java support
Other Requirements	N/A

Table 3: Component Summary

convenience, we make use of the Sesame framework[7] to materialize the facts, and build the resource graph on top of the RDFStore introduced in Sesame. More details on the specifications of Sesame RDF stores and RDF model can be in [12].

The contextualization process is triggered every time a resource in the active system has to be archived, since the corresponding context information has to be archived as well. Given an input resource id, the Contextualizer extracts a subset of the RDF graph and returns it as the context information for the input resource. The method pursued to identify the context information depends on how the reasoner sub-component is implemented. For this first prototype, a simple contextualization is performed by extracting the neighbours of a given resource which are far from it up to k-links, where k is an open parameter which might be tuned through user experiments. As previously stated, it is envisioned that more complex contextualization strategies will both take the content of the resource into account and enrich the graph by adding implicit resource relationships detected through reasoning.

Summarizing, from an input-output perspective, the component takes a resource id as input and returns a set of RDF facts, stored in JSON format, as context information for the resource. It will be released either as a command line tool or as a RESTful service. Within the ForgetIT framework, the component will be queried whenever a resource has to be contextualized: the related context information will be extracted and stored along

[7] `http://www.openRDF.org`

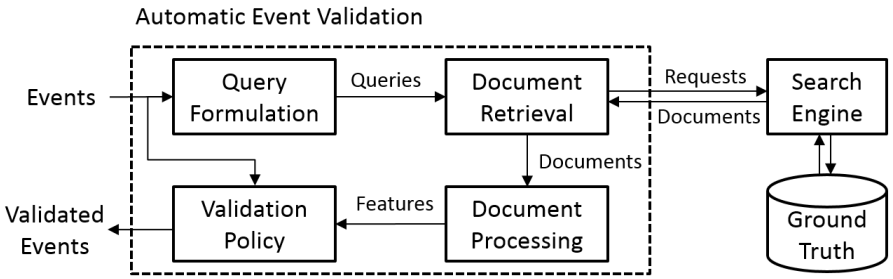

Figure 3: High-level architecture of the automatic event validation.

with the resource itself.

3.4 Event-based Contextualization

3.4.1 Overview

Under the assumption that events related to the content or to the time of a document can help to contextualize it, we want to stress the importance of events, or entity relationships, within the contextualization process. This component works at the end of the process: events are assumed to be already detected by a given event detection tool, and the goal consists in validating them, i.e. determining whether those events are actually true or not.

Event detection algorithms, which generally infer the occurrence of real-world events from natural language text, are continuously being designed and they always require a ground truth for their validation. However, the lack of an annotated and comprehensive ground truth makes the evaluation onerous for humans, who have to manually search for events inside it. With this component, we envision to automatize the evaluation process. The component validates events by estimating the temporal relationships among their representative entities within documents in the Web. The underling approach is going to be published in the 36-th European Conference on Information Retrieval [13].

The required input for this prototype component consists in a set of events, usually being the output of a particular event detection algorithm. Every event is described as a set of entities that participated to it, along with a start and end date representing the time period in which the event was supposed to occur.

The high-level architecture of the component is depicted in Figure 3. For every event, the *query formulation* component generates one or more queries by taking into account the participating entities, the starting time, and the ending time. Then, the *document retrieval* component interacts with the *ground truth* by feeding the queries into a Web search engine (Bing, in the current implementation) and retrieving the results, i.e. documents. Note that, in order to make queries, the ground truth is supposed to be indexed. The *document processing* component extracts features (entities and temporal expressions) from

the retrieved documents, which are used by the *validation policy* to validate the event.

The output of the component is, for every event, a value between 0 and 1 representing the confidence for the event to be true. These values are computed by the validation policy which, for every (event, document) pair, takes into account how many event entities are present in the document and which is their position with respect to ”valid” temporal expressions, i.e. dates in the document that are included in the event time span.

3.4.2 Usage Instructions

Functional description	Event validation
Input	textual file
Output	textual file
Limitations/Scalability	Rule-based validation policy
Language/technologies	Java
Hardware Requirements	N/A
OS Requirements	Any OS with Java support
Other Requirements	N/A

Table 4: Component Summary

The input events are fed into the component in form of a text file, where each line represents an event and contains both the entities and the time period associated to it. However, the input format might be easily changed to be a XML file. Output confidence values are stored in a textual file but, as for the input, they can easily stored as a XML file.

The component would fit into the ForgetIT framework by working in the middleware and serving any kind of event validation request. For instance, events detected and extracted from documents by the Extractor component could be further validated before being stored and processed for contextualization. The component will be released either as a command line tool or as a RESTful service.

4 Prototype Component for the Contextualization of Images

4.1 Overview

Multimedia content, e.g. an initial image collection (seed content), can be significantly enriched and contextualized by looking at other content that is semantically related to the former. Such content could be, for example, images or image collections that were captured during the same or a similar event as the seed content, or landscape images captured possibly at different times but in the same location. That is, for a given image collection, similar collections can be retrieved; then, images of the latter collections that convey additional related information, e.g. images covering different aspects of the same event, can be added to the seed collection. Thus, the similar collections can augment the information the given image collection conveys and contribute to better putting the initial images in context.

Thus, for contextualizing a given image collection, the initial ForgetIT approach consists of a) finding similar image collections (respecting of course any user privacy preferences) and then b) adding contextual information in the form of selected images contained in these collections. Realizing this pipeline is possible with the use of WP4-developed analysis tools and further techniques being developed in WP6. Specifically, for each image collection, high-level representation of it (model vectors) can be extracted using the concept detection tools being developed in WP4, and collection summarization through clustering can also be performed, again using WP4 techniques. Collection similarity can then be performed in WP6 by calculating the Euclidean or Mahalanobis distance between the collections' images or cluster centres, using the model vectors as features. Subsequently, exploiting the collections that are found to be similar, contextual information can be added to the seed collection (most notably, additional images, but also metadata that are associated with any of the similar collections that have been found). Besides visual or concept-level similarity evaluation, geographical and time information can be additionally employed (if they are available) in order to better assess the similarity between image collections. Additionally, the same basic idea can be applied when using textual information for further contextualizing an image collection: information that can be reliably extracted from related text but is often not easily understandable by looking at an image (e.g. time, location, people names) can be added to an image collection in order to put the visual information in context. Of course, the specific techniques that will be used in such a case for assessing e.g. the similarity between an image collection and text excerpts are different from those used for evaluating the similarity between visual information only, but this is something that we will also address in WP6 at a later stage.

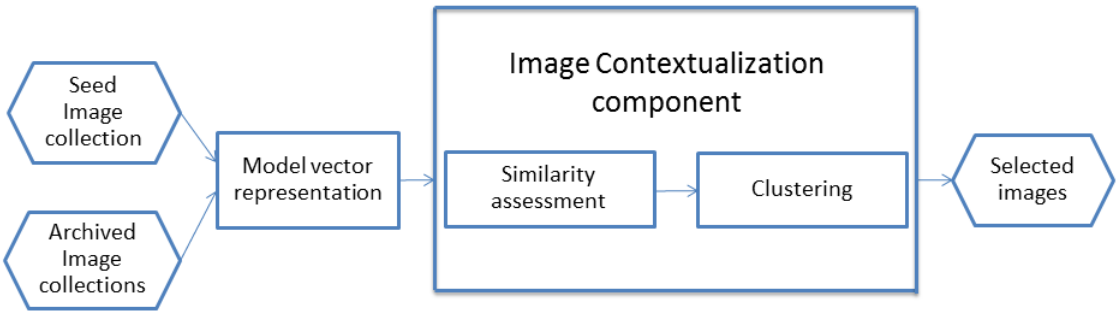

Figure 4: Image Contextualization pipeline

4.2 Usage Instructions

In Figure 4, the image contextualization approach is shown. The user passes an image collection (seed image collection) as input to the component in order for the collection to be contextualized. Several other image collections are already stored into the ForgetIT archive and are considered as the additional image collections that will be examined for augmenting the seed collection. However, the user is able to pre-select a subset of them in order to accelerate the entire procedure.

The already stored image collections should be represented with model vectors (output of concept detection procedure, see section 5 of D4.2 for more details). Therefore, the target collection should be also passed through a concept detection procedure. Each image is represented by a model vector with N-elements, where N is the number of concepts that are considered.

The contextualization method consists of the steps described below:

- The seed collection is clustered. For the clustering, k-means algorithm is employed in which image model vectors are used as input data (see section 8 of D4.2 for more details). The number of clusters can be set according to the Rule of thumb[8] pursuant to which the number of clusters equals the square root of the half of the number of set elements.
- The maximum (Md) of the distances between all cluster centres is calculated. Md is a measure of the diameter of the seed collection set.
- The centre of all the collections is calculated and then the Euclidean distance between all the centres of the archived image collections and the seed one is calculated. The archived collections that are closer than $c_1 \cdot Md$ to the seed collection, are considered as similar collections to the seed one.

[8] http://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set

- For each similar collection, clustering is performed using the same procedure with the one followed for the seed collection clustering. For each cluster, the Euclidean distances between itself and all the cluster centres of the seed collection is calculated. The minimum of these distances (md) used as a measure of cluster proximity of the similar collection to the seed collection. Since we wish to exclude the clusters that are almost identical with the seed collection, we reject the clusters for which $md < c_2 \cdot Md$. For all the clusters of the similar collections for which $md \geq c_2 \cdot Md$, we calculate the image that is closest to the cluster centre and export it as contextual image to the seed collection.

It should be noted that the clustering method used above has been described in detailed and realized in WP4 (see section 8 of D4.2 for more details).

The thresholds c_1 and c_2 that are used in the algorithm above were initially chosen by experimentation. In the next few months, we will evaluate this first approach to contextualization of image collections and, based on the evaluation results, we will revise the approach as needed and also work on more elaborate ways for selecting the value of any parameter such as c_1, c_2 that will be involved.

A Matlab function has been created that realizes the proposed procedure. The `visual_contexualization_ForgetIT` function is called as: `visual_contexualization_ForgetIT(seedCollection, additionalCollections)`

The input arguments of the function are: `seedCollection` which is the seed image collection directory and `additionalCollections` that is a text file that lists the other collections directories. It should be noted that each directory should contain a file with the model vectors (output of concept detection procedure). The method saves in a specific directory its result, i.e. the set of additional images that can augment the seed collection.

A summary of the technical details of the image contextualization software is shown in Table 5. The prototype component described in this section can be downloaded from `http://www.forgetit-project.eu/en/downloads/workpackage-6/`. Please contact the project for access to this protected section of the website.

Functional description	Image collection contextualization
Input	Seed image collection and the additional ones
Output	A directory containing the selected images
Limitations/Scalability	None
Language/technologies	Matlab
Hardware Requirements	N/A
OS Requirements	Windows OS
Other Requirements	Concept detection and clustering tools being developed in WP4

Table 5: Image contextualization software summary

5 Conclusions

This deliverable has continued the work from D6.1 by presenting an in-depth discussion and formal model of the ForgetIT approach to contextualization. This model has helped to inform the development of the current contextualization components prototypes as well as highlighting areas for future research. This deliverable is, however, focused upon the first round of prototype contextualization components which have been developed within the work package.

Five contextualization components from three project partners have been described and made available via this deliverable. These components cover a wide range of issues faced by contextualization and aim to process both images and text as well as attempting to re-contextualize documents which may have been archived without a suitable context.

The next phase of development will see these components integrated into the ForgetIT framework, as soon as feasible, as well as the further development of these, and possibly new, components. Thought will also turn to the detailed evaluation of these components in order for us to acquire a deeper understanding of how real users form a context and if the components fully fulfil their requirements.

References

[1] Nattiya Kanhabua and Kjetil Nørvåg. Exploiting Time-Based Synonyms in Searching Document Archives. In *Proc. 10th Annual Joint Conf. on Digital Libraries*, pages 79–88, 2010.

[2] Nina Tahmasebi, Gerhard Gossen, Nattiya Kanhabua, Helge Holzmann, and Thomas Risse. NEER: An Unsupervised Method for Named Entity Evolution Recognition. In *Proc. 24th Int. Conf. on Computational Linguistics*, pages 2553–2568. ACL, 2012.

[3] Danica Damljanovic and Kalina Bontcheva. Named Entity Disambiguation using Linked Data. In *Proceedings of the 9th Extended Semantic Web Conference*, 2012.

[4] Niraj Aswani, Genevieve Gorrell, Kalina Bontcheva, and Johann Petrak. Multilingual, ontology-based information extraction from stream media. Technical Report D2.2.2, TrendMiner Project Deliverable, 2013.

[5] M. Sahlgren. An introduction to random indexing. In *Proc. of the Methods and Applications of Semantic Indexing Workshop*, Copenhagen, Denmark, 2005.

[6] D. Milne and I. H. Witten. Learning to link with Wikipedia. In *Proc. of the 17th Conf. on Information and Knowledge Management (CIKM)*, pages 509–518, 2008.

[7] D. Rao, P. McNamee, and M. Dredze. Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, Multi-lingual Information Extraction and Summarization*. 2011.

[8] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009.

[9] Wei Gao, Peng Li, and Kareem Darwish. Joint topic modeling for event summarization across news and social media streams. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1173–1182, New York, NY, USA, 2012. ACM.

[10] Nattiya Kanhabua, Roi Blanco, and Michael Matthews. Ranking related news predictions. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 755–764, New York, NY, USA, 2011. ACM.

[11] Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009.

[12] Programming with sesame, January 2013.

[13] Andrea Ceroni and Marco Fisichella. Towards an entity–based automatic event validation. In *ECIR*, 2014.