Hybrid Acquisition of Temporal Scopes for RDF Data

Anisa Rula[1], Matteo Palmonari[1], Axel-Cyrille Ngonga Ngomo[2], Daniel Gerber[2], Jens Lehmann[2], and Lorenz Bühmann[2]

[1] University of Milano-Bicocca
{anisa.rula|palmonari}@disco.unimib.it
[2] Universität Leipzig, Institut für Informatik, AKSW
{ngonga|dgerber|lehmann|buehmann}@informatik.uni-leipzig.de

Abstract. Information on the temporal interval of validity for facts described by RDF triples plays an important role in a large number of applications. Yet, most of the knowledge bases available on the Web of Data do not provide such information in an explicit manner. In this paper, we present a generic approach which addresses this drawback by inserting temporal information into knowledge bases. Our approach combines two types of information to associate RDF triples with time intervals. First, it relies on temporal information gathered from the document Web by an extension of the fact validation framework DeFacto. Second, it harnesses the time information contained in knowledge bases. This knowledge is combined within a three-step approach which comprises the steps matching, selection and merging. We evaluate our approach against a corpus of facts gathered from Yago2 by using DBpedia and Freebase as input and different parameter settings for the underlying algorithms. Our results suggest that we can detect temporal information for facts from DBpedia with an F-measure of up to 70%.

Keywords: #eswc2014Rula, Temporal Information Extraction, Temporal Semantic Web, Temporal Scoping, Fact Checking

1 Introduction

Over the last few years, the Linked Open Data (LOD) Cloud has developed into a large amalgamation of diverse data sets from several domains [2]. Some of these data sets provide encyclopedic knowledge on the real world. For example, DBpedia [12] contains RDF extracted from the infoboxes of Wikipedia.[3] While some of the statements contained in the LOD Cloud are universally valid (e.g., the fact that the birth place of Mario Balotelli is Palermo), a large portion of the facts which are referred to by the triples in the LOD Cloud are only valid within a certain time interval, which we call their *time scope*. For example, DBpedia states that Mario Balotelli plays for the teams Inter Milan and Manchester City. While the semantics of the predicate `dbo:team`[4] remains a matter of discussion,

[3] `http://wikipedia.org`
[4] `dbo` stands for `http://dbpedia.org/ontology/`.

manifold applications such as question answering [20], temporal reasoning and temporal information retrieval [9] require having the temporal scope of facts such as "Mario Balotelli plays for the team Inter Milan from 2007 to 2010".

In this paper, we introduce an approach for detecting the temporal scope of facts referred to by triples (short: the temporal scope of the triples). Given a fact (i.e., an RDF triple), our approach aims to detect the time points at which the temporal scope of the triple begins and ends. Two sources can be envisaged for gathering such information: the document Web and the Linked Data Web. Our approach is able to take advantage of both: the document Web is made use of by extending upon a fact validation approach [11], which allows detecting Web documents which corroborate a triple. In contrast to typical search engines, the system does not just search for textual occurrences of parts of the statement, but tries to find webpages which contain the actual statement phrased in natural language. The second source of information for time scopes is the Web of Data itself. Here, we use the RDF data sets that contain the facts, e.g., DBpedia and Freebase, for possible time scopes and devise an algorithm for combining the results extracted from Web documents with those fetched from RDF sources. The algorithm consists of three main steps. First, the evidence extracted from Web documents is *matched* against a set of relevant time intervals to obtain a significance score for each interval. Second, a small set of more significant intervals is *selected*. Finally, the selected intervals are *merged*, when possible, by considering their mutual temporal relations. The set of disconnected intervals [1] returned by the algorithm defines the temporal scope of the fact. We also propose two *normalization* strategies that can be applied to the data extracted from Web documents before running the algorithm, to account for the significance of dates appearing in the documents corroborating the input fact.

The main contributions of this paper are:

- We introduce a temporal extension of the DeFacto framework based on a sliding window approach on fact-confirming documents.
- We present an approach for modeling a space of relevant time intervals for a fact starting from dates extracted from RDF triples.
- We devise a three-phase algorithm for temporal scoping, i.e. for mapping facts to sets of time intervals, which integrates the previous steps via matching, selection and merging.
- Finally, we evaluate the integrated algorithm on facts extracted from DBpedia and Freebase against the Yago2 knowledge base.

The rest of this paper is structured as follows: Section 2 describes our general approach and the system infrastructure. In Section 3, we describe how temporal information is extracted from web pages using a temporal extension of the DeFacto algorithm [11]. Section 4 shows how this information can be mapped to a set of time intervals specifying its temporal scope. We then evaluate the approach by using temporal scopes from Yago2 as gold standard and facts from DBpedia and Freebase as input in Section 5. We give an overview of related work in Section 6. Finally, we conclude and give pointers to future work.

2 Problem Definition and Approach Overview

Linked Open Data describes resources identified by HTTP URIs by representing their properties and links to other resources using the RDF language. Given an infinite set $\mathcal{U}$ of URIs, an infinite set $\mathcal{B}$ of blank nodes, and an infinite set $\mathcal{L}$ of literals, a statement $< s, p, o > \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ is called an *RDF triple*. As the use of blank nodes is discouraged for LOD [3], we will assume that the subject and the property are URIs, while the object can be either a URI or a literal.

Most of the resources described in the LOD Cloud represent real-world objects (e.g., soccer players, places or teams); we use the term *entities* as a short form for *named individuals* as defined in the OWL 2 specification. According to the LOD principles [3], we assume that each entity e can be dereferenced. The result of the dereferencing is an RDF document denoted d^e which represents a *description* of the entity e. We say that d^e *describes* e and d^e is an *entity document* [8]. As an example, an entity document d^e returned by DBpedia in NTriples [6] format contains all the RDF triples where e occurs as a subject. In this work, RDF triples occurring in entity documents are called *facts*.

A *fact* represents a relation between the subject and the object of the triple and, intuitively, it is considered true when the relation is acknowledged to hold. We use the term *volatile facts* to refer to facts that change over time, and are represented by triples whose validity can be associated with a temporal context (e.g., <Balotelli, team, Inter_Milan> refers to a fact occurring from 2007 to 2010). Adopting a terminology used in previous work on temporal information extraction, we call *temporal scope* of facts the specification of the time during which facts occured [19].

Despite several models to represent time into RDF having been suggested, only a small amount of RDF data sets annotate triples with their temporal scope. This is partly due to the sophisticated meta-modeling strategies needed to represent temporal annotations in RDF [18]. As a consequence, several knowledge bases contain *volatile facts* without explicitly annotating the triples with information about their temporal scope. We define a *temporal annotation* of a fact a couple $< f, [t_i, t_j] >$, where f is a fact and $[t_i, t_j]$ is a time interval delimited by a starting time point t_i and an ending time point t_j. In this paper we regard time as a discrete, linearly ordered domain, as proposed in [7]. In our discrete time model, two intervals $[t_i, t_j]$ and $[t_h, t_k]$ are *disconnected* iff $t_j < t_h$, or $t_k < t_i$, and *connected* otherwise. The *temporal scope* of a fact f is defined as a set of - possibly many - temporal annotations of f with *disconnected* time intervals.

The problem addressed in this paper can be defined as follows: for each volatile fact $f \in F$ extracted from a data set Δ, we map f to a set $TS^f = \{[t_{i_1}, t_{j_1}], ..., [t_{i_n}, t_{j_n}]\}$ where TS^f defines the temporal scopes of f and each element represents a time interval when the fact is true. Figure 1 gives an overview of our solution. Evidence is extracted from Web and RDF documents and a space of possible time intervals relevant to the fact is built; the evidence extracted from Web documents is matched against the space of relevant time intervals and a final set of temporal scopes are associated with the input fact.

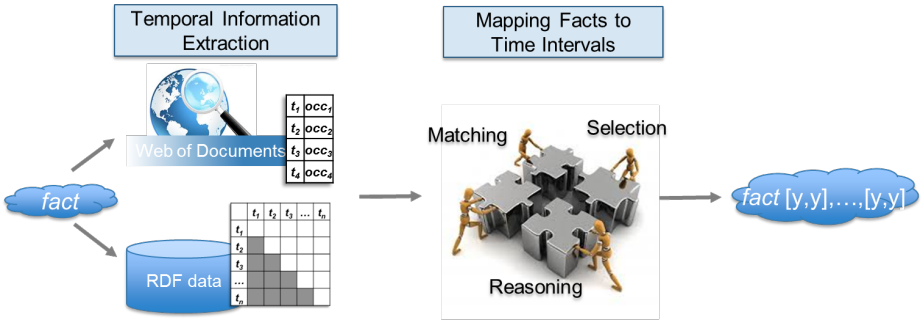


Fig. 1. Approach Overview

3 Temporal Information Extraction

In this section we describe the methods we adopted to extract temporal information from two sources: the Web of documents and the Web of Data. The latter source contains the facts to be assigned with a temporal scope.

3.1 Extraction of Temporal Information from the Web

Temporal DeFacto is an extension to the DeFacto framework presented in [11]. The system takes an RDF triple as input and returns a confidence value for this triple as well as possible evidence for the fact. The evidence consists of a set of webpages, textual excerpts from those pages and meta-information on the pages. The first task of DeFacto is to retrieve webpages which are relevant for the given task. The retrieval is carried out by issuing several queries to a search engine. These queries are computed by verbalizing the RDF triple using natural-language patterns extracted by the BOA framework [5]. As a next step, the highest ranked webpages for each query are retrieved, which are candidates for being sources for the input fact. Both the search engine queries as well as the retrieval of webpages are executed in parallel to keep the response time for users within a reasonable limit. Once a webpage has been retrieved, we extract plain text by removing HTML markup and apply our fact confirmation approach on this text. In essence, the algorithm decides whether the web page contains natural language formulations of the input fact. If no webpage confirms a fact according to DeFacto, then the system falls back on light-weight NLP techniques and computes whether the webpage does at least provide useful evidence. In addition to fact confirmation, the system computes different indicators for the trustworthiness of a webpage as presented in [15]. These indicators are of central importance because a single trustworthy webpage confirming a fact may be a more useful source than several webpages with low trustworthiness. In addition to finding and displaying useful sources, DeFacto also outputs a general confidence value for the input fact. This confidence value ranges between [0, 1] and serves as an indicator for the user: Higher values indicate that the found sources appear to confirm the fact and can be trusted. Low values mean that

not much evidence for the fact could be found on the Web and that the websites that do confirm the fact (if such exist) only display low trustworthiness. The generated provenance output can also be saved directly as RDF and abides by the PROV Ontology[5]. The source code of the DeFacto algorithms and DeFacto's user interface are open-source[6].

Temporal Extension of DeFacto. To also incorporate time information into the fact validation process we extended DeFacto as follows. On all retrieved webpages we apply the Stanford Named Entity Tagger[7] and extract all entities of the *Date* class. We then examine all occurrences $occ_{so} \in Occ_{so}$ of the subject and object label of the input fact (or their surface forms, e.g. "Manchester United F.C." might also be called "ManU") in a proximity of less than 20 tokens. We then build a local context window of n characters before and after occ_{so} and analyze all contained *Date* entities. Finally we return a distribution of all dates and their number of occurrences in a given context. Hence, the output of temporal DeFacto for a fact f <*s, p, o*> can be regarded as a vector DFV over all possible time points t_i whose i^{th} entry is the number of co-occurrences of s or o with t_i. We will use the function $dfv_i(f, t_i)$ to denote the value of DFV_i for the fact f.

3.2 Extraction of Temporal Information from RDF documents

Given a set F of facts to map to time intervals, we first identify the set of entities E that occur as subjects for the set of facts in F. Given the entity e subject of the fact, we use the HTTP content negotiation mechanism to retrieve the entity document d^e. As an example, given the fact <*Cristiano Ronaldo, team, Real Madrid*>, we extract the RDF document describing *Cristiano Ronaldo*. Once an entity document has been retrieved, we extract time points from the temporal triples that are contained in the entity document. We define a *temporal triple* a triple of the form $< s, p, t >$, where the object t is a time point. As an example, although DBpedia does not provide temporal annotations for the fact <*Cristiano Ronaldo, team, Manchester United*>, it has the temporal triples <*Cristiano Ronaldo, years, 2009*> and <*Cristiano Ronaldo, youthYears, 1995*>. Some of these dates refer to other facts of the same entity; however, the link between the facts containing the dates and the facts these dates were related to has been lost. We use temporal triples available in the knowledge base under the assumption that this information can be *relevant* to define the scope of facts.

Given an entity e subject of a fact, we identify temporal triples in the entity document d^e and extract dates by using regular expressions, which identify standard date formats and variations. In this step we adopt an approach that was used in previous work [18]. We add to this set of extracted dates a date representing the *current time*. As a result of this step, each fact $f \in F$ is associated with a set of *relevant time points* T^e extracted from the RDF document

[5] `http://www.w3.org/2011/prov/`
[6] `https://github.com/AKSW/DeFacto`
[7] `http://www-nlp.stanford.edu/software/CRF-NER.shtml`

describing the subject of the fact. In principle, our approach can consider dates represented at any granularity level; in the following examples and in the experiments, time is represented at the year level similarly as in other related work [13, 19].

Intuitively, we want to use the relevant time points T^e associated with an entity e to identify a set of most *relevant time intervals* for scoping facts having e as subject; in this way, we can reduce the space of all possible time intervals considered for an individual fact. The set of time intervals relevant to an entity e is defined as the set of all time intervals whose starting and ending points are members of T^e. Relevant time intervals are represented using an upper triangular matrix, i.e., a square matrix where all entries below the diagonal are 0.

Given a set T^e of relevant time points for an entity e, a relevant time interval matrix (*Relevant Interval Matrix* for short) RIM^e is an upper triangular matrix of size $|T^e| \times |T^e|$ defined as follows:

$$RIM^e = \begin{bmatrix} rim^e_{t_1,t_1} & \cdots & \cdots & rim^e_{t_1,t_n} \\ 0 & \ddots & & \\ 0 & 0 & \ddots & \\ 0 & 0 & 0 & rim^e_{t_n,t_n} \end{bmatrix} \quad (1)$$

Columns and rows of a relevant interval matrix RIM^e for an entity e are indexed by ordered time points in T^e; each cell $rim^e_{t_i,t_j}$ with $i, j > 0$ represents the time interval $[t_i, t_j]$, where $t_i, t_j \in T^e$. At the moment we assign a placeholder value `null` to each cell $rim^e_{i,j}$ such that $i \leq j$. In the matching phase, we will use entity-level RIMs as schemes for fact-level matrices; in these fact-level matrices `null` values will be replaced by scores that represent the significance of intervals for individual facts. Observe that the use of an upper triangular matrix is suitable for representing time intervals since the time intervals represented in the cells in the lower part of the matrix ($i > j$) are not valid by definition. Also note that, the cells in the diagonal of the RIM^e matrix represent time intervals whose start and end points coincide.

4 Mapping Facts to Time Intervals

The process used to provide a final mapping between a volatile fact and a set of time intervals defining its temporal scope consists of three phases: 1) Temporal Distribution-to-Time Intervals Matching, 2) Time Intervals Selection 3) Time Interval Merging. Figure 2 shows an overview of the application of the three phases to a fact f, with a RIM built from a set of four relevant time points extracted from the entity document. The algorithm can take as input the vectors returned by Temporal DeFacto, i.e., DFVs, as well vectors normalized using two functions defined in Section 4.2.

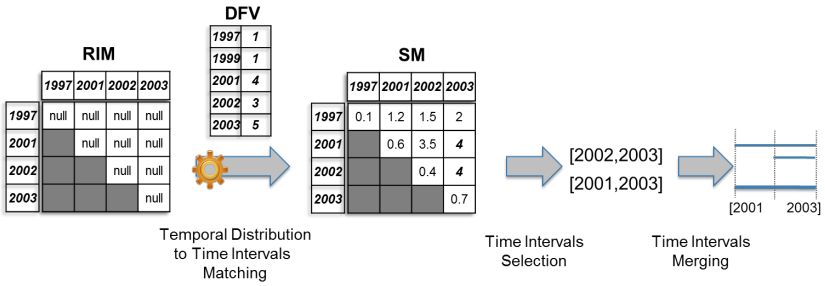


Fig. 2. Time Interval Mapping Overview

4.1 Matching, Selection and Reasoning

Temporal Distribution to Time Intervals Matching. The inputs of the matching phase for a fact f that has an entity e as subject are the following: a relevant interval matrix RIM^e extracted the entity document d^e and a time distribution vector $DFV^{e,f}$. Probabilistic time distribution vectors obtained by normalization (see Section 4.2) can be also used as input instead of DFVs. The matching phase returns an *interval-to-fact significance matrix*, Significance Matrix (SM) for short, $SM^{e,f}$ associated with the fact f. An $SM^{e,f}$ is a triangular square matrix having the same size and structure of the input RIM^e. As a next step, `null` values of a RIM^e are replaced with significance scores returned by a matching function.

In practice, to build an $SM^{e,f}$ of a fact f with subject e, we match a fact-level DFV^f associated to the fact f against an entity-level RIM^e, i.e. the matching aims to inject a time distribution vector into RIM^e by producing a significance matrix $SM^{e,f}$. The matching function $match(DFV^f, RIM^e) = SM^{e,f}$, where e is an entity and f is a fact, is given as follows:

$$sm_{i,j} = \begin{cases} 0 & \text{if } rim_{i,j} = 0 \\ \frac{\sum_{k=i}^{j} dfv(f,k)}{(j-i)+1} & \text{if } rim_{i,j} = null \wedge i < j \\ dfv(f,i) * w_{i,j} & \text{if } rim_{i,j} = null \wedge i = j \end{cases} \tag{2}$$

Since the denominator $(j-i)+1$ in the formula used in case two represents the number of time points included in the interval $[i,j]$, the formula is equal to the average of DFVs for the time points contained in the interval. As an example, the score for a cell $sm_{1995,2000}$ is defined as the average value of DFV for the time points between 1995 and 2000 (including the starting and ending points). Since the elements in the diagonal have length equal to 1, the formula used in case three is equivalent to multiplying the score computed with the formula used in case two for a weight $w_{i,j}$; we use this weight to penalize the scores in the diagonal as we discovered that formula in case two would assign high scores to the element in the diagonal, thus favoring time intervals with length equal to 1 in the selection phase. Intuitively we want to penalize elements in the diagonal

unless they are the only significant values selectable in the SM matrix. The weight is defined as inversely proportional to the difference between the length of the considered interval (equal to 1) and $length(DFV^f)$ the length of the DFV vector as follows:

$$w_{i,j} = \frac{1}{c * length(DFV^f)} \tag{3}$$

where c is a constant used to control the score reduction ratio applied to the elements in the diagonal of the SM matrices.

Mapping Selection. Once we have a set of significance matrices $SM^{e,f_1}, ..., SM^{e,f_n}$, each one associated with a fact f_i referred to e, we then select the time intervals that might be mapped to the considered facts. We propose two basic selection functions that use SMs; both functions can select more than one interval to associate with a fact f. The *top-k* function selects the k intervals that have best scores in the SM matrix. The *neighbor-x* selects a set of intervals whose significance score is close to the maximum significance score in the SM matrix, up to a certain threshold. In other terms, we define the *neighborhood of the time interval with maximum significance score* as the set of intervals whose significance scores fall in the range defined by the maximum score as upper bound and by a threshold based on a parameter x as lower bound. The threshold is linearly proportional to the maximum significance score, so that the threshold is higher when the maximum significance is higher (e.g., 0.9) and lower when the maximum significance is lower. The parametric function *neighbor-x* with an SM and a parameter x given as input is defined as follows:

$$neighbor(SM, x) = \left\{ [i, j] \mid sm_{i,j} \geq maxScore - \frac{x * maxScore}{100} \right\} \tag{4}$$

The two basic functions *top-k* and *neighbor-x* can be combined into a function *neighbor-k-x* that selects the top-k intervals in the neighborhood of the interval with higher significance score. Observe that neighbor-0 is equal to top-1 for every value of the parameter x. The neighbor-k function behaves as a filter on the results of the top-k function, by selecting only intervals whose significance is close enough to the most significant interval.

Interval Merging via Reasoning. Finally, we use rules based on Allen's interval algebra to merge the selected time intervals and map each fact to a set of disconnected intervals. Let a and b be two time intervals associated with a fact f and defined respectively by $[t_i, t_j]$ and $[t_h, t_k]$; we merge a and b into an interval defined by $[min(t_i, t_h), max(t_j, t_k)]$ whenever one of the following conditions, each one based on Allen's algebra relations [1], is verified:

- a overlap b or a is-overlapped-by b
- a meets b or a is-met-by b
- a during b or b during a
- a starts b or b starts a
- a finishes b or b finishes a

The temporal scope of a fact is defined by the set of disconnected time intervals mapped to it after the interval merging phase.

4.2 Temporal Distribution Normalization

Two types of normalization functions can be envisaged: *local normalization* and *global normalization*. These functions aim to transform the output vector of temporal DeFacto (the DFV vector) into a probabilistic time distribution (PTD) vector. Here, the main idea of the local normalization is that the PTD contains the probability that the fact $<s,\ p,\ o'>$ should be mapped to a given time point t_i. The main drawback of such a normalization is that it does not take the PTD vector for other facts $<s,\ p,\ o'>$ into consideration. We thus defined global normalization functions that allow transforming the output of temporal DeFacto for all triples with subject s and predicate p. When normalization strategies are adopted, the PTDs are used instead of DFVs in Equation 2.

Local Normalization. Several approaches can be used to generate a PTD. The approach we follow is based on the frequency-based interpretation of the output of Temporal DeFacto: The i^{th} entry in DFV basically states the number of times f co-occurred with the time point t_i in a relevant document. Thus, the probability that f co-occurs with the time point t_i is:

$$PTD_i = \frac{DFV_i}{\sum_{j=1}^{|T^e|} DFV_j}. \tag{5}$$

Global Normalization. Our approach to the computation of a global normalization was based on χ^2 statistics. Given a resource s, a predicate p and a point t_i in time, the aim of the normalization was to compute the significance of the value of DFV_i. Let E_i be the expected value of DFV_i for the time t_i, computed as average value of all DFV_i entries for the resource s over all objects of p. The significance of the time t_i for the triple $< s,\ p,\ o_j >$ with vector DFV is then

$$\frac{(DFV_i - E_i)^2}{E_i}. \tag{6}$$

5 Experimental Evaluation

5.1 Experimental Setup

Methodology and Gold Standard. To evaluate our approach we acquire the temporal scopes of a population of volatile facts from three different domains and compare the results of our method against a gold standard. We use manually curated data from Yago2[8] as gold standard. We omitted all facts with `null` values, i.e. missing starting or end time. We choose Yago2 because it is one of the few large open-source knowledge bases that provides temporal annotations for a significant number of facts (714,925 time points associated with facts).

[8] `http://www.mpi-inf.mpg.de/yago-naga/yago/`

Significant parts of DBpedia[9], Freebase[10] and Yago2 are extracted from the same source which makes it possible to automatically map some facts in DBpedia or Freebase to facts in Yago2. We therefore use facts in DBpedia, and Freebase in our experiments and we extract RDF data from these sources. We additionally consider the case where RIMs (see Section 3.2) are created with the time points returned by Temporal DeFacto, to simulate the case when temporal information from RDF data is not available.

Properties of Interests. The facts considered in our experiments are defined using the top three properties having the largest number of occurrences in Yago2. Table 1 shows the properties and the number of facts for each property.

Table 1. Properties of interest and the number of facts for each property

Property	Number of facts
<ismarriedTo>	3,501
<holdsPoliticalPosition>	5,610
<playsFor>	114,367

Because we have a limit of queries sent through temporal DeFacto, which is imposed by traffic limitations of its underlying search engine, we perform the experiment on a subset of all available facts by applying some selection rules: the top 1000 facts on the most important soccer players who are born after 1983 ($\leq$30 years old), the top 1000 facts on politicians born after 1940, and the top 500 facts on celebrities born after 1930.

Measures. In order to evaluate the accuracy of our method, we measured the degree to which the temporal scope we retrieved is correct w.r.t. the gold standard. Therefore, for each fact, we consider the degree of overlap between the retrieved intervals and the interval in the gold standard. This degree of overlap can be computed by adapting the well-known metrics of precision, recall and F_1-measure to this problem leveraging the discrete time model. Intuitively, the precision of a temporal scope can be measured by the number of time points in the temporal scope generated by our solution that fall into the time interval in the gold standard. The recall of our solution can be measured by the number of time points in the gold standard that are covered by the temporal scope.

Let $R(f)$ be the set of time points in the temporal scopes retrieved for a fact f and Ref(f) be the set of time points included in the reference temporal scopes for f; the following formulas capture the intuitions described above:

$$\mathit{precision}(f) = \frac{|R(f) \cap \mathit{Ref}(f)|}{|R(f)|}, \quad \mathit{recall}(f) = \frac{|R(f) \cap \mathit{Ref}(f)|}{|\mathit{Ref}(f)|}. \tag{7}$$

Precision and recall for a fact f can be combined as usual in F_1-measure defined as the harmonic mean of precision and recall. Note that: when $\mathit{precision}(f) = 1$, each interval in the retrieved temporal scope is included in the interval of the

[9] `http://dbpedia.org/`
[10] `http://freebase.com/`

gold standard; when $recall(f) = 1$, all the time points in the interval of the gold standard are covered by the retrieved temporal scopes; when $F_1(f) = 1$ the temporal scope contains exactly the same time points as the gold standard.

Baseline. Given that no prior algorithm aims to tackle exactly the task at hand, we computed the precision, recall and F-measure that a random approach would achieve. To this end, we assumed that given the restrictions we set on the intervals within which our solutions must lie (e.g., 1983-2014 for soccer players), a random solution would simply guess for each date whether it should be part of the final solution. This serves as a lower bound for the score a temporal scoping algorithm should achieve.

5.2 Results and Discussion

In order to evaluate the overall accuracy of scoping facts with temporal intervals we need to set up different configurations for each component of each phase. Hence, we approximate the best configurations for some key components of the proposed approach by using genetic programming[11] based on opt4j[12], an open-source framework comprising a set of optimization algorithms. Genetic programming allows to determine an appropriate configuration of our approach. In the configuration setup we consider the interval selection functions and the merging process of the selected intervals through reasoning (see Section 4.1) as well as the normalizations strategies applied to the Temporal DeFacto Vectors to obtain Probabilistic Temporal Distributions (PTDs) (see Section 4.2).

In the first experiment, we compare the best configurations for properties of interests, i.e., (1) `isMarriedTo`, (2) `holdsPoliticalPosition` and (3) `playsFor`. The space of relevant time intervals (RIM) is built from time points collected from three different sources, i.e., Temporal DeFacto, Freebase and DBpedia. Table 2 reports for each property and for each source the best F-measure achieved

Table 2. Results of best configurations for all property of interests.

Property	Baseline		Temp DeFacto			Freebase			DBpedia		
	#facts	F_1	Config	#facts	F_1	Config	#facts	F_1	Config	#facts	F_1
1	500	0.163	top-3	311	**0.511**	top-1 loc	213	0.477	top-1 loc	264	0.505
2	1000	0.263	top-3	709	0.586	neigh-10-2	242	0.549	neigh-10	702	**0.699**
3	1000	0.207	top-3	709	0.545	neigh-10	524	0.547	neigh-10	705	**0.600**

by our approach. In one case, the RIM and the scores are defined with evidence retrieved only from the web of documents (TempDeFacto for short). In other two cases, the RIM is build with dates extracted from the web of data (Freebase or DBpedia) and the scores are computed by injecting evidence from the web of documents into this matrix. We observe that our approach perform much better than the baseline, which does not use a prior algorithm, for every property and

[11] `http://goo.gl/2ve3xP`
[12] `http://opt4j.sourceforge.net/`

for every source used to construct the RIMs. The best configurations is obtained for the property `holdsPoliticalPosition` with time points extracted from DBpedia and with selection function neighbor-k with $x = 10$. The configuration that extracts time points from DBpedia outperforms Freebase and Temporal DeFacto results except for the property `isMarriedTo`. The reason for this major gain can be explained with the quantity and quality of relevant time points extracted from the three sources. The problem is that Freebase and Temporal DeFacto do not provide enough time points which can prevent the effective identification of intervals. We notice that, while local normalization improves the results in one experiment (for the property `isMarriedTo`), the global normalization strategy is never optimal in any experiment. We will now compare the reasoning and selection functions.

Different Components. Table 3 shows the contribution of reasoning for the best configurations identified in the previous experiment. We use the full approach with and without reasoning and apply it on the three properties. We observe that enabling reasoning improves the performance of the temporal scoping of facts. This validates our motivation behind using Allen's Algebra, as it can get rid of incomplete intervals.

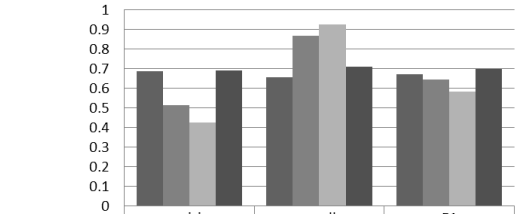
Table 3. Effect of using reasoning during temporal scoping from the three best configurations.

			With reasoning		Without reasoning	
Property	Source	Config	# facts	F_1	# facts	F_1
1	Temp DeFacto	top-3	311	0.511	505	0.467
2	DBpedia	neigh-10	702	0.699	822	0.667
3	DBpedia	neigh-10	705	0.60	977	0.563

Based on these results, we can evaluate the effect of selection functions and their application in DBpedia for the property `holdsPoliticalPosition`. Figure 3 compares four configurations. We observe that recall is improved when k is increased but on the other side precision decreases as the approach returns larger intervals including the correct interval and additional incorrect time points. The best precision-recall is given with the combined selection function, neighbor-k with $x = 10$.

6 Related Work

The work presented in this paper relies on two areas of research: the extraction of time information and fact checking. **Extraction of Time Intervals.** Several machine learning approaches have been developed to discover links between events and temporal information (e.g., dates) into one or more sentences of a document where the event is mentioned [21]. In alternative, the work in [10] presents a method to link events or facts with timestamps according to a



	precision	recall	F1
top-k, k = 1	0.686	0.654	0.67
top-k, k = 2	0.515	0.865	0.645
top-k, k = 3	0.426	0.924	0.583
neighbor x = 10	0.689	0.709	0.699

Fig. 3. Effect of using selection function during temporal scoping of `holdsPoliticalPosition` from DBpedia source.

classification approach. In contrast to these approaches, our approach is completely unsupervised and it does not need training data. Temporal Information Extraction (TIE) [13] is a more recent system that finds a maximal set of temporal annotations for events mentioned in a given sentence. Therewith, it can infer relations between these events using the temporal annotations. Instead of Allen-style intervals [1], TIE uses time points. However, this approach is not sufficient to extrapolate the temporal scope of facts because it focuses on the micro-reading of temporal annotations in single documents or sentences. Although the aim of temporal bounding [4] and our approach is the same since both retrieve temporal constraints given a fact, there are fundamental differences. NLP techniques employed in temporal bounding are more sophisticated but at the same time more expensive and extract evidence from the text on a limited corpus. Our approach uses softer, but more efficient, NLP techniques to extract evidence from the whole web. Moreover, our approach investigates how to complement evidence retrieve from texts with evidence from the web of data.

Timely Yago2 [9] has the objective of enriching facts with temporal scopes. Instead of using the original data source (i.e, Wikipedia) where the link between facts and time intervals is explicitly made available, our approach exploit the evidence from the web of data where facts are not associated with time intervals and the web of documents, i.e., free text evidence. Yago2 identify the time of a fact if the time of the entities occurring in the fact is known and the property occurring in the fact belongs to a predefined category. PRAVDA [22] is a recently proposed method to harvest basic and temporal facts from free text. The approach is based on a semi-supervised label propagation algorithm that determines the similarity between structured facts and textual facts. Yet, it does not use the verbalization of RDF triples to check for RDF triples in text like DeFacto does. The system CoTS provided in [19] is similar to our system since it also detects temporal scopes for facts. In contrast to our approach, CoTS relies on document meta-data such as its creation data to assign temporal scopes to facts. To ensure that it gathers enough information, CoTS aggregates evidences

from a large number of documents to temporally scope a set of facts. This approach is complementary to our current approach and can easily be combined with it.

Fact Checking. Regarding the fact checking part of our approach, a very recent algorithm was developed in [14]. It describes an approach, which allows to evaluate the truth value of statements by querying the web and processing unstructured web pages. It is based on training a supervised classifier with features extracted from web pages. A difference to our own previous work on DeFacto [11] is that we optimised the extraction by considering a larger variety of features related to patterns found on websites and also combined those features with an analysis of the trustworthiness of web pages. In another line of research on fact checking in [16, 17], trustworthy is also a central element. The authors rely on a model based on hubs and authorities. This model allows to compute the trustworthiness of facts and websites by generating a k-partite network of pages and facts and propagating trustworthiness information across it. The approach returns a score for the trustworthiness of each fact. An older yet similar approach is that presented in [23]. Here, the idea is to use a 3-partite network of webpages, facts and objects and apply a propagation algorithm to compute weights for facts and webpages.

7 Conclusion and Future Work

In this paper, we presented an approach for mapping volatile facts to time intervals. Our approach is hybrid and combines information from the document Web and temporal statements included in knowledge bases. We evaluated our approach on volatile facts extracted from DBpedia and Freebase by using cleaned-up temporal scopes extracted from Yago2. The cleaning was made necessary by approximately 50% of the information in that knowledge base being either incomplete or inconsistent (begin after end). This underlines the *difficulty of the task at hand*. Our approach achieved promising results, delivering approximately 70% F-measure on the facts at hand. In future work, we will create a larger gold standard for evaluating temporal scopes. Finally, we will develop applications that use temporal information. For example, we plan to develop a temporal extension of the TBSL question answering framework that can answer questions such as "When did Balotelli play for Inter Milan?".

Acknowledgements

This research has been supported in part by FP7/2013-2015 COMSODE (under contract number FP7-ICT-611358).

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] S. Auer, J. Lehmann, and A.-C. N. Ngomo. Introduction to linked data and its lifecycle on the web. In *5th RR*, pages 1–75, 2011.
[3] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *IJSWIS*, pages 1–22, 2009.
[4] L. Derczynski and R. Gaizauskas. Information retrieval for temporal bounding. In *4th ICTIR*, pages 29:129–29:130. ACM, 2013.
[5] D. Gerber and A.-C. N. Ngomo. Extracting Multilingual Natural-Language Patterns for RDF Predicates. In *18th EKAW*. Springer, 2012.
[6] J. Grant and D. Becket. Rdf test cases - N-Triples. Technical report, W3C Recommendation, 2004.
[7] C. Gutiérrez, C. A. Hurtado, and A. A. Vaisman. Temporal RDF. In *2nd ESWC*, pages 93–107, 2005.
[8] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.
[9] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.
[10] D. Hovy, J. Fan, A. Gliozzo, S. Patwardhan, and C. Welty. When did that happen?: linking events and relations to timestamps. In *13th EACL*, 2012.
[11] J. Lehmann, D. Gerber, M. Morsey, and A.-C. Ngonga Ngomo. DeFacto - deep fact validation. In *11th ISWC*, pages 312–327. Springer, 2012.
[12] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *SWJ*, 2014.
[13] X. Ling and D. S. Weld. Temporal information extraction. In *25th AAAI*, 2010.
[14] M. B. Mehdi Samadi, Manuela Veloso. OpenEval: Web information query evaluation. In *27th AAAI*, 2013.
[15] S. Nakamura, S. Konishi, A. Jatowt, H. Ohshima, H. Kondo, T. Tezuka, S. Oyama, and K. Tanaka. Trustworthiness analysis of web search results. In *11th ECDL*, 2007.
[16] J. Pasternack and D. Roth. Generalized fact-finding. In *20th WWW*, 2011.
[17] J. Pasternack and D. Roth. Making better informed trust decisions with generalized fact-finding. In *20th IJCAI*, 2011.
[18] A. Rula, M. Palmonari, A. Harth, S. Stadtmüller, and A. Maurino. On the diversity and availability of temporal information in linked open data. In *11th ISWC*, 2012.
[19] P. P. Talukdar, D. T. Wijaya, and T. Mitchell. Coupled temporal scoping of relational facts. In *5th WSDM*, pages 73–82, 2012.
[20] C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, and P. Cimiano. Sparql template-based question answering. In *21st WWW*, 2012.
[21] N. UzZaman and J. F. Allen. Trips and trios system for tempeval-2: Extracting temporal information from text. In *SemEval*, pages 276–283. ACL, 2010.
[22] Y. Wang, M. Dylla, Z. Ren, M. Spaniol, and G. Weikum. Pravda-live: interactive knowledge harvesting. In *21st CIKM*, pages 2674–2676. ACM, 2012.
[23] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *IEEE Trans. Knowl. Data Eng.*, 20(6):796–808, 2008.