Universität des Saarlandes
Max-Planck-Institut für Informatik
AG5

Extraction of Temporal Facts and Events from Wikipedia

Masterarbeit im Fach Informatik
Master's Thesis in Computer Science
von / by

Erdal Kuzey

angefertigt unter der Leitung und betreut von / supervised and advised by

Prof. Dr. Gerhard Weikum

begutachtet von / reviewers

Dr. Martin Theobald
Second Reviewer

April 2011

Hilfsmittelerklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Non-plagiarism Statement

I hereby confirm that this thesis is my own work and that I have documented all sources used.

Saarbrücken, den 04. April 2011,

(Erdal Kuzey)

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlichtwird.

Declaration of Consent

Herewith I agree that my thesis will be made available through the library of the Computer Science Department.

Saarbrücken, den 04. April 2011,

(Erdal Kuzey)

To all members of Kuzey family...

"Eternity is a very long time, especially towards the end."

Stephen Hawking.

Acknowledgements

First and foremost, I am deeply grateful to my mother and father for their great support. I would not be where I am today without their love and encouragement.

I would like to express my sincere gratitude to my advisor and supervisor Prof. Dr. Gerhard Weikum for giving me the opportunity to complete my thesis successfully. His guidance and support during my thesis were invaluable and motivated me for pursuing this thesis.

I am also very thankful to my colloborator Yafang Wang for all her help.

A special note of thanks to Yagiz Kargin and Christina Teflioudi for their invaluable friendship, support and encouragement. I would also like to thank all students of 411 lab, especially Niket Tandon and Tuan Anh Tran for the motivating and enjoyable environment they provided.

Finally, I would like to thank IMPRS-CS for the assistance and financial support that I received from them. It gave me total freedom to work on my master thesis.

Abstract

In recent years, the great success of Wikipedia and the progress in information extraction techniques led to automatic construction of large scale knowledge bases which have Subject-Predicate-Object style facts extracted from both semi-structured and natural language text of Wikipedia articles. Those knowledge bases consist of millions of entities, relations about them and their semantic types. Unfortunately, most of the current knowledge bases focus on static facts and ignore their temporal dimension, although, the vast majority of facts are evolving with time and are valid during a particular time period.

In this thesis, we introduce a complete information extraction framework which harvests temporal facts and events from semi-structured and free text of Wikipedia articles to enrich a temporal ontology (T-YAGO). Furthermore, this thesis discusses methods for introducing a temporal dimension to time-agnostic knowledge bases. In addition, several experiments and evaluations are presented to show the effectiveness of the methods proposed.

Contents

Acknowledgements ix

Abstract xi

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Problem Statement . . . 1
- 1.3 Contributions . . . 2
- 1.4 Outline . . . 3

2 Background and Related Work 5
- 2.1 Knowledge Bases and Ontologies . . . 5
- 2.2 Information Extraction . . . 6
  - 2.2.1 Wikipedia-based IE . . . 7
- 2.3 Temporal Information and Fact Extraction . . . 8
  - 2.3.1 Temporal Expressions . . . 8
  - 2.3.2 TimeML and TARSQI . . . 9

3 Data Model 13
- 3.1 Resource Description Framework-RDF . . . 13
- 3.2 The YAGO Model . . . 15
  - 3.2.1 Introduction . . . 15
  - 3.2.2 Knowledge Representation Model . . . 16
- 3.3 The T-YAGO Model . . . 18

4 Temporal Fact and Event Extraction from Semi-structured Text 21
- 4.1 Definition of T-fact and Event . . . 21
- 4.2 Extraction from Infoboxes . . . 23
- 4.3 Extraction from Categories . . . 26
- 4.4 Conclusion . . . 30

5 Temporal Fact and Event Extraction from Free Text 33
- 5.1 Pattern-based Extraction by Normalizing explicit Dates . . . 34
- 5.2 Dictionary-Based Extraction by Normalizing Implicit Dates . . . 38
  - 5.2.1 Mapping Temporal Phrases to T-facts . . . 40
    - 5.2.1.1 Identification of Temporal Expressions . . . 41
    - 5.2.1.2 Interpretation . . . 42

5.2.1.3 Normalization 45
5.2.2 Connecting T-facts to Known Base Facts 46
5.3 Conclusion 48

6 Experiments and Evaluation **49**
6.1 Evaluation Measures 49
6.1.1 Precision and Recall 49
6.2 Extraction from Semi Structured Text 50
6.2.1 Data Collection 51
6.2.2 Harvesting Infoboxes 51
6.2.3 Harvesting Categories 52
6.3 Extraction from Free Text 55
6.3.1 Data Collection and Experimental Settings 55
6.3.2 Pattern-Based Extraction by Normalizing Exact Dates 56
6.3.3 Dictionary-Based Extraction by Normalizing Implicit Dates 56
6.3.4 Summary 60

7 Conclusions and Future Directions **61**
7.1 Conclusion 61
7.2 Future Directions 61

List of Figures **62**

List of Tables **65**

A Classification of Temporal Expressions **67**

Bibliography **71**

Chapter 1

Introduction

1.1 Motivation

In recent years, the great success of Wikipedia and the progress in information extraction techniques led to automatic construction of large scale knowledge bases which have Subject-Predicate-Object style facts extracted from both semi-structured and natural language text of Wikipedia articles. DBpedia [ABK+07], KnowItAll[ECD+04b], Intelligence In Wikipedia [WWA+08], and YAGO [SKW07a] can be given as significant examples of such knowledge bases. Those machine readable knowledge bases consist of millions of entities, relations about them and their semantic types. Unfortunately, most of the current knowledge bases focus on static facts and ignore their temporal dimension, although, the vast majority of facts are evolving with time and are valid during a particular time period. For instance, the fact *Jacques Chirac* `holdsPoliticalPosition` *President of France* is less helpful then the fact *Jacques Chirac* `holdsPoliticalPosition` *President of France* with the temporal scope "17-05-1995, 16-05-2007".

Introducing a temporal dimension to knowledge bases will lead to many rich applications, such as visualizing timelines of important people or events, querying the knowledge base for a certain time interval, capturing the ordering of closeness or relatedness of important events. Such a rich knowledge base will be a great asset for historians, media analysts, social researchers, etc.

1.2 Problem Statement

Extraction of static facts is different than extracting temporal facts, since ontological temporal facts are higher order facts (facts about facts), whereas, static facts mostly

stand in binary relations. For instance, the fact *Jacques Chirac* `holdsPoliticalPosition` *President of France* can be created from free text once the entities *Jacques Chirac, President of France* are detected and the semantic relation between them is correctly mapped to the pre-defined relation `holdsPoliticalPosition`. However, detecting the correct time period about previous fact, and connecting it to the fact is much harder than constructing the previous fact, since the temporal scope of previous fact might occur in many different forms. Moreover, the granularity of date expressions, relative time expressions, and adverbial phrases makes our job more difficult.

Our goal is to enrich a temporal ontology (T-YAGO)[1] via using advanced temporal fact extraction techniques from semi-structured parts of Wikipedia. We further exploit T-YAGO as a dictionary to bootstrap the extraction from free text of Wikipedia articles.

1.3 Contributions

We make the following contributions:

1. We define a complete methodology to extract temporal facts from Wikipedia articles, so as to create (or enrich) a temporal ontology.
2. We develop an information extraction framework which extracts temporal facts and events from semi-structured parts of Wikipedia.
3. We introduce an automatic pattern induction method to create patterns and to rank them based on their frequency statistics.
4. We presented a pattern-based extraction technique to harvest temporal facts from free text by normalizing explicit dates.
5. We utilize our temporal knowledge base in order to normalize implicit temporal phrases in free text, which we name as dictionary based temporal information extraction.
6. We present a novel method to introduce a temporal dimension to already existing facts.
7. We finally provide experimental results showing the efficiency of our approaches.

[1] T-YAGO is probably the first work in the line of temporal fact extraction. It extends the YAGO ontology by introducing new predicates to capture the temporal aspects of existing facts. T-YAGO contains around 300K temporal facts extracted from semi-structured part of Wikipedia.

1.4 Outline

The rest of this thesis is organized as follows: In Chapter 2, we give background information and discuss the related work. In Chapter 3, we present the data model used in T-YAGO. Chapter 4 describes the extraction of temporal facts from semi-structured parts of Wikipedia articles. In Chapter 5, we introduce our pattern-based and dictionary-based extraction frameworks. Chapter 6 describes the experiments we used for evaluating our approaches and their results. Finally, in Chapter 7, we summarize our work and propose directions for future work.

Chapter 2

Background and Related Work

2.1 Knowledge Bases and Ontologies

A knowledge base is a special kind of database for knowledge management, providing the means for the computerized collection, organization, and retrieval of knowledge.[1] Knowledge bases contain a set of data, often in the form of rules that describe the knowledge in a logically consistent manner.

An ontology is a formal representation of knowledge as a set of concepts and the relationships between those concepts. Informally, ontologies are the structural frameworks for the organization of information. An ontology can define the structure of the data of a knowledge base.

Why an Ontology Is Important?

As the knowledge is formally described in ontologies, ambiguity of concepts is avoided since entities and classes are unique. In addition to that, all ontologies use a formal language with specific vocabulary, which enables reasoning. Moreover, ontologies enable the reuse of the domain knowledge. For example, different domains may need to represent the notion of time. This representation includes the notions of time intervals, time points, granularity of time, measures of time, and so on. If there is an ontology which represents the notion of time in details, then other ontologies can simply reuse it.

There are domain-specific ontologies which are restricted to a certain topic [BCS06, HL93, Ont08], and also domain-independent ontologies (upper ontologies) which contain general knowledge about the world [Mil95, Len95, NP01].

[1] http://en.wikipedia.org/wiki/Knowledge_base

Most of the widely employed ontologies are man-made such as [Mil95, Len95, NP01] requiring huge amount of manual effort and domain experts. Recently, the idea of automatically creating ontologies from web sources is emerging. The ontologies created in this way have an acceptable accuracy and contain millions of facts [SKW07a, HSB+, ABK+07, WW07, WW08]. These ontologies use advanced information extraction methods to gather high amount of precise facts.

2.2 Information Extraction

Information extraction is the subdiscipline of artificial intelligence that tries to extract information from semi-structured or un-structured machine readable documents and store it in a structured way that can be queried directly. Traditionally, information extraction is associated with template based extraction from natural language documents, which was a popular task of the *Message Understanding Conferences (MUC)* [Sun92]. MUC was the first large scale effort to encourage research into automatic information extraction (IE). During Message Understanding Conferences, theregradually arose set of typical IE tasks:

- **Named entity recognition:** This is probably one of the most popular IE task which aims to recognize person names, organizations, locations, date, time, numbers, money and percentages.
- **Event extraction:** The goal of event extraction is recognizing events, their participants and their settings. It goes further by linking the individual events in a story line.
- **Coreference resolution:** This aims to detect coreference and anaphoric links between the entities in text. Informally, it aims to determine whether two expressions in natural language text refer to the same entity, person, time, place and event.
- **Relation extraction:** The task of relation extraction tries to identify relations between entities.

From an ontological point of view, information extraction serves for fact extraction. Unlike manual approaches such as, [Mil95, Len95, NP01], information extraction methods try to gather facts from text documents in an automatic manner. During the process of fact extraction, IE approaches use a wide variety of models and methods, including linguistic, machine learning and rule-based (template-based) approaches [Sar08]. These

approaches usually start with a given set of target relations and try to extract as many facts belonging to the relations as possible via bootstrapping the extraction with example seeds.

DIPRE [Bri99], KnowItAll [ECD+04b], Snowball and its variations [AG00, LNYW10, ZNL+09] are among the most renowned projects of this type. They harvest manually specified seed facts (example facts) of a given relation in order to find textual patterns that can possibly express the relation. Then the best patterns are statistically determined and applied on the text to find new facts from occurrences of these patterns. LEILA [SIW06] improved this method via using counter-examples in addition to example facts as seeds. TextRunner [YCB+07] is even more ambitious to extract all instances of all meaningful relations from Web documents, a task called as *Open IE* or *Explorative IE*. SOFIE [SSW09] combines the pattern-based IE methods of LEILA with entity disambiguation features and ontological consistency constraints to harvest ontological facts with high precision. PROSPERA [NTW11] goes one step further via introducing notion of ngram-itemsets for richer patterns which are statistically weighted.

2.2.1 Wikipedia-based IE

Recently, Wikipedia has attracted the attention of information extraction community, due to its high quality and unique structure. Wikipedia has several attributes that make it very convenient for information extraction:

URI: Wikipedia assigns a unique identifier for each concept. The URI of an article only maps to one concept.

Infoboxes: Wikipedia articles have special table-style sections, called infoboxes, which summarize the important points of an article.

Categories: Each article has a category section which provides membership information for the article. An article may have more than one category. Categories are also grouped into super-categories, forming in this way a hierarchical network.

Disambiguation pages: A disambiguation page provides alternative meanings for an ambiguous title. Disambiguation pages are valuable resources for the entity disambiguation task that we have mentioned earlier.

Redirect pages: A redirect page has no content itself, but is a pointer for one term to another article. Redirect pages are used for synonymy detection.

In this section we have discussed about IE centric approaches to create ontologies. DBpedia [ABK+07] and YAGO [SKW07a] extensively exploit semi-structured part of

Wikipedia articles (infoboxes and categories) to construct full-fledged ontologies. Kylin and Kylin Ontology Generator [WW07, WW08] do not only use semi-structured parts of Wikipedia but also use freetext of Wikipedia articles to create an ontology.

2.3 Temporal Information and Fact Extraction

Research on information extraction aims to harvest facts from either semi-structured or unstructured text, such as from Web documents. However, all of the IE work that we discussed till now focus on static facts encoded as binary relations. In real world, most of the facts are time-dependent, which results in a new information extraction area, called temporal fact extraction. Although the idea of temporal information extraction is still new, equipping facts with temporal information has already attracted lots of attention. Especially, predominantly temporal sources such as, news archives or Wikipeda create a fruitful research area for temporal fact extraction.

The closest work on temporal fact extraction is the event extraction task introduced in TempEval Challenge [VGS+07] workshops and TimeML [PV09]. It basically focuses on tasks which involve identifying event-time and event-event temporal relations. In TempEval Challenge, only a restricted set of Allen-style [All83] temporal relations are used. The state-of-the-art system in the field of temporal information extraction is TARSQI [VP08] which succeeded in detecting temporal expressions by marking events and generating temporal associations between times and events. Another system called TIE [LW10] did not only detect the temporal relationships between times and events, but also used probabilistic inference so as to bound the time points of the beginning and the end of each event.

2.3.1 Temporal Expressions

Extracting temporal information from documents requires deep understanding of temporal expressions and their structure. In this section, we present an overview of temporal expressions.

In [AGBY07], the authors distinguish three classes of temporal expressions:

Explicit: Explicit temporal expressions have an immediate interpretation such as "*23 February 1985*", *December 2003*, *1997*, etc.

Implicit: Implicit temporal expressions require a background knowledge (a dictionary) to be interpreted. "*Christmas of 2001*", *Valentine Day of 2002*, etc. are examples for implicit temporal expressions.

Relative: These include temporal expressions which require a reference point in order to be interpreted such as, "*yesterday*", *last year*, *next week*, etc.

Although this classification seem to be enough, we extend it via adding the notion of "*temporal phrases*" which is different than temporal adverbs, such as "*last year*", *after two years*, etc.

Temporal phrase: In our case, we define a temporal phrase which contains special events that has to be interpreted, such as "*during French Revolution*", *after the Battle of Waterloo*, *before death of mayor Achille Peretti*, etc. The interpretation of such phrases requires a deep background knowledge. We solve this problem via creating a large temporal knowledge base which knows events and temporal facts, and then employing this knowledge base as a dictionary to interpret temporal phrases.

The extraction of temporal expressions can be separated into an identification phase and an interpretation phase. In the identification phase, parts of the text that constitute the temporal expressions are identified. In the interpretation phase, the meaning of the identified temporal expressions is determined by mapping them onto the timeline. For relative temporal expressions this includes determining the right temporal anchor and resolving the temporal expression relative to this anchor. Notice that the separation into these two phases is only conceptual - actual tools may interleave the two phases or put more intermediate phases.

2.3.2 TimeML and TARSQI

The state of the art tool for extracting temporal expressions is TARSQI [VP08] which uses TimeML [PV09] specification language. TIMEML is a markup language to annotate temporal expressions and event expressions in free text. TimeML annotates temporal expressions by decorating them with the `TIMEX3` tag. Figure 2.1 shows an excerpt from a news article which is annotated by TARSQI[2].

TimeML distinguishes different types of temporal expressions such as `DURATION, DATE, SET, TIME`. TARSQI uses the current time as reference time, unless it is provided with the publication time of the document. The normalized value of a temporal expression is shown by the `VAL` attribute.

[2]Figure is taken from [Ber10]

In order to have a better understanding of temporal expressions, we conducted a deep analysis about temporal expression types, resulting in a classification of temporal expressions, their types, TimeML support, and TARSQI support. This classification can be found in Appendix A.

It is important to note that the concept of *event* in the above systems or tasks is different from our concept of *fact* or *event*. Rather than checking relationships of base facts, systems like TARSQI or TIE focus on extracting time information either by time-events or by event-events. Here, events are defined as verb forms with tenses, adjectives, and nouns that describe the temporal aspects of the events. In the example of "Obama accepted the award in Oslo last year.", systems like TARSQI consider the verb "accepted" to be an event, and adverbial phrase "last year" to be a time point. And they explore the relation between "accepted" and "last year", so-called event-time relation. This task however is very different from our notion of events and temporal facts referring to entities and relations, which is detailly explained in details in Chapter 4.

```
Hong Kong is poised to hold the first election in more
than half <TIMEX3 tid="t3" TYPE="DURATION" VAL="P100Y">a
century</TIMEX3> that includes a democracy advocate
seeking high office in territory controlled by the Chinese
government in Beijing.  A pro-democracy politician,
Alan Leong, announced <TIMEX3 tid="t4" TYPE="DATE"
VAL="20070131">Wednesday</TIMEX3> that he had obtained
enough nominations to appear on the ballot to become the
territory's next chief executive.  But he acknowledged
that he had no chance of beating the Beijing-backed
incumbent, Donald Tsang, who is seeking re-election.
Under electoral rules imposed by Chinese officials,
only 796 people on the election committee -- the bulk
of them with close ties to mainland China -- will be
allowed to vote in the <TIMEX3 tid="t5" TYPE="DATE"
VAL="20070325">March 25</TIMEX3> election.  It will be
the first contested election for chief executive since
Britain returned Hong Kong to China in <TIMEX3 tid="t6"
TYPE="DATE" VAL="1997">1997</TIMEX3>.  Mr.  Tsang, an able
administrator who took office during the early stages of
a sharp economic upturn in <TIMEX3 tid="t7" TYPE="DATE"
VAL="2005">2005</TIMEX3>, is popular with the general
public.  Polls consistently indicate that three-fifths
of Hong Kong's people approve of the job he has been
doing .  It is of course a foregone conclusion -- Donald
Tsang will be elected and will hold office for <TIMEX3
tid="t9" beginPoint="t0" endPoint="t8" TYPE="DURATION"
VAL="P5Y">another five years</TIMEX3>, said Mr.  Leong,
the former chairman of the Hong Kong Bar Association.
```

FIGURE 2.1: New York Times article annotated by using TARSQI

Chapter 3

Data Model

In our work, we need to store extracted temporal facts in a machine-readable knowledge base which describes the knowledge in a logically consistent manner.

Actually, this is studied under the field of Artificial Intelligence called knowledge representation. Knowledge representation aims to form the knowledge in a way that facilitates reasoning and inferencing from knowledge. Currently, the most common knowledge representation model is *Resource Description Framework (RDF)* which is the underlying model in many ontological knowledge bases. We used an RDF-style knowledge representation model which we name *T-YAGO model*. T-YAGO model extends the YAGO model by introducing temporal concepts.

In this section, we first introduce the original RDF model and its variants and then explain the YAGO model and its differences from RDF. Finally, we will introduce T-YAGO model which can also represent temporal concepts in an ontology.

3.1 Resource Description Framework-RDF

RDF was designed for describing relationships between resources, using specific vocabularies, so that knowledge models we have in the real world can be better correlated for information re-use. RDF uses statements about resources in the form of *subject-predicite-object* expressions. These expressions are called *SPO triples* in RDF terminology, or *facts* in ontology terminology. The subject denotes the resource, and the predicate expresses a relationship between the subject and the object. For example, one way to represent the fact "*Albert Einstein is a German scientist*" in RDF is as a triple with:

- a subject denoting "*Albert Einstein*": `Albert_Einstein`

- a predicate representing "*is a*" : `isA`
- and an object denoting "*German scientist*": `German_Scientist`

One of the most important aspects of RDF is that everything (people, locations, events, artifacts, concepts, etc.) is uniquely represented by a resource. Each resource is identified by a *Uniform Resource Identifier (URI).*

URI: A URI is a string which follows a certain syntax specification to represent resources, or entities from an ontological point of view. A particular URI can refer to one and only one entity, whereas an entity can be referred by multiple URIs. For example, http://mpii.de/yago/Albert_Einstein is the URI identifying the entity "*Albert_Einstein*".

In order to denote facts, RDF uses statements between entities, which can be seen as binary assertions or relations about entities.

Statement: An RDF statement is an assertion expressing that two a subject and an object stand in a binary relation represented by a predicate. For example, to state that Albert Einstein was born in Ulm as an ontological fact, we write the following statement:

- subject:`http://mpii.de/yago/Albert_Einstein`
- predicate:`http://mpii.de/yago/wasBornIn`
- object:`http://mpii.de/yago/Ulm`

For the sake of simplicity, we can omit the URI of the entities when representing statements. For example, the previous statement can be also represented as:

- `Albert_Einstein` *wasBornIn* `Ulm`

For the rest of this thesis, we will omit the URI form of the entities. Note that the object of a statement can also be a literal instead of an entity.

Literal: A literal is a string that represents a value, which may belong to a particular data type. For instance, the string "*1985*" represents the date value 1985. Thus, the fact "*Albert Einstein was born on 14-03-1879*" can be represented as:

- `Albert_Einstein` *wasBornOnDate* ''*14-03-1879*''

Another property that RDF provides us with is to reify statements, ie., to make statements about statements. Formally, each statement is assigned a URI and is treated as a resource, about which additional statements can be made. For instance, assume that we want to provide additional information about the fact `Albert_Einstein` *wasBornIn* `Ulm` and we want to say that it is extracted from Wikipedia rather than an arbitrary web source. First, we should create the URI `AlbertEinsteinBornUlm` for the fact `Albert_Einstein` *wasBornIn* `Ulm`, then we can reify it as:

- `AlbertEinsteinBornUlm` *wasExtractedFrom* `Wikipedia`.

RDFS: RDF was extended by *Resource Description Framework Schema* with new specifications and vocabularies to strengthen the expressiveness of RDF. RDFS[BG04] allows super classes for entities as well as enables literals to have certain data types.

For the sake of clarity, hereafter we use noun in plain format to indicate entities, verb phrases in italic to indicate relation names, and quoted strings to indicate literals.

3.2 The YAGO Model

3.2.1 Introduction

YAGO is a large ontology, which is automatically generated by information extraction techniques [Suc]. It knows 2 million entities, such as famous people, cities, historical events, movies, companies, etc. and knows around 20 million facts about these entities. The knowledge of YAGO is extracted from Wikipedia. YAGO has a systematic approach to harvest knowledge from infoboxes and categories of Wikipedia articles, in order to generate a vast amount of highly precise facts. The YAGO extractors use declarative information extraction approaches by defining rules and heuristics to harvest infoboxes and categories. YAGO is reported to have around 95% precision which is almost the same as human quality. In addition to infoboxes and categories, YAGO exploits the redirect and disambiguation pages of Wikipedia in order to get information about several meanings and abbreviations of entity names.

YAGO links the information extracted from Wikipedia to the information from WordNet [Mil95], in order to create a clean ontological taxonomy. Therefore, each entity is guaranteed to have a type and each relation is guaranteed to have domain and range types, which is exploited for a quality control mechanism for facts. YAGO uses *canonicalization* to make each fact and each entity unique in the entire ontology. It further

ensures *type checking* to eliminate individuals which do not have a pre-defined YAGO class. Type checking also verifies the plausability of newly extracted facts which do not obey the domain and range constraints of their relation.

3.2.2 Knowledge Representation Model

YAGO introduces a novel knowledge representation model which builds on RDFS. It extends RDFS by giving more importance on reification of statements. The YAGO model identifies concepts and objects in the knowledge base as entities which have unique identifiers, just as RDFS does. However, instead of using URIs for describing resources (entities), YAGO uses simple, local identifiers. It adopts the SPO style of RDF, in order to represent entities standing in a binary relation. Furthermore, the YAGO model assigns fact identifiers to every fact. Thus, each fact can stand in a higher order relation, i.e., any fact can be subject of another fact via its fact identifier. However, RDFS requires creating a new URI for a fact which is to be reified. The fact identifiers of YAGO, simply literals of type Integer, are created automatically during the extraction process and are locally stored in the knowledge base. The reification of a YAGO fact is shown below:

- `#12` `Albert_Einstein` *hasWonPrize* `Nobel_Prize_in_Physics`
- `#13` `#12` *happenedOnDate* ‘‘*1921*’’

where `#12` and `#13` represent the fact identifiers of particular facts.

In addition to *common entities (classes and all individuals)* and fact identifiers, relations are also classified as entities in the YAGO model. The YAGO relations have pre-defined names and semantics.

Formally, a YAGO ontology can be described as a *reification graph* which has entities as nodes and edges as relations. Edges do not only connect two nodes, but also can connect a node and an edge or even two edges. Figure 3.1 shows an excerpt of YAGO RDF graph, where the red nodes represent classes, blue nodes represent individuals and labeled edges represent relations between entities.

Data Types

Since RDFS uses the data types defined by XML schema, which are machine-oriented and not always semantically intuitive [BM+01], YAGO introduces its own data types. YAGO groups all data types, such as *Number, String, TimeInterval, Quantity, etc.* into the class *literal*. These literal classes have subclasses. For example, the *TimeInterval*

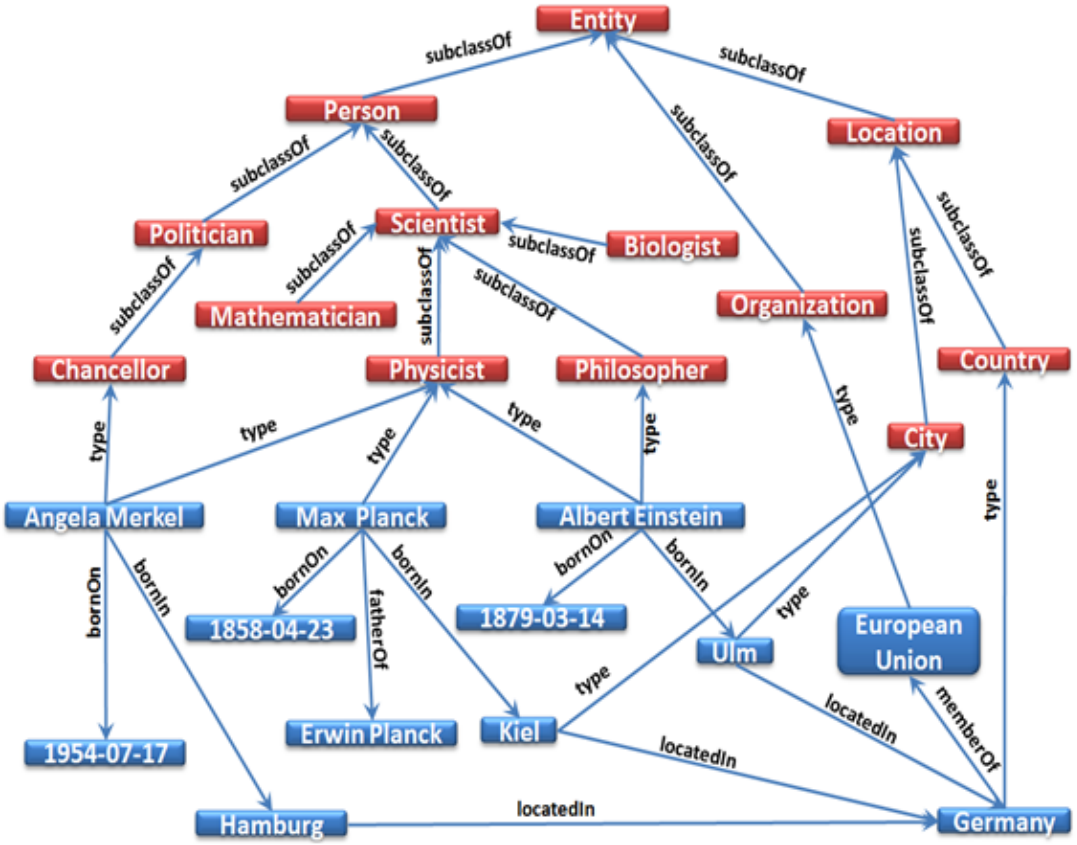

FIGURE 3.1: An excerpt of YAGO RDF graph.

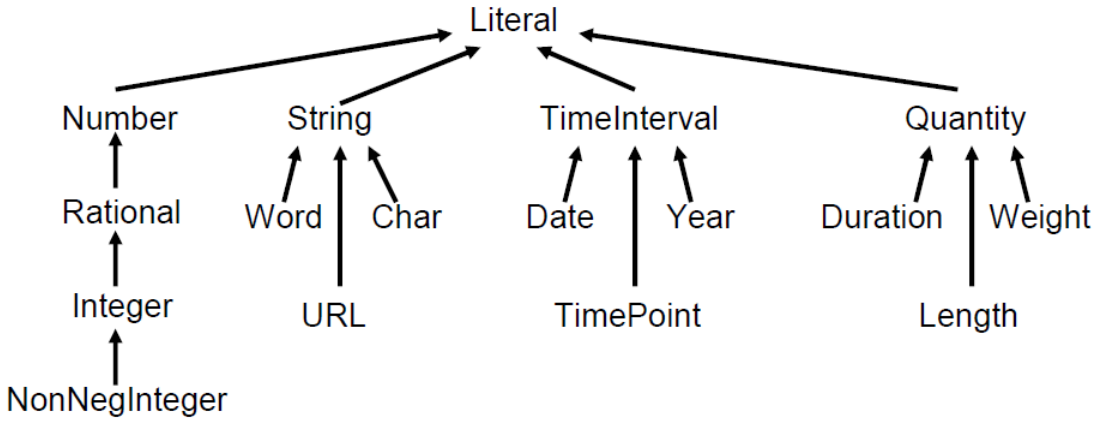

FIGURE 3.2: YAGO data types.

literal class has *Date, TimePoint,* and *Year* as subclasses. YAGO literals can be seen in Figure 3.2. [1]

[1]Figure is taken from [Suc].

3.3 The T-YAGO Model

The T-YAGO knowledge representation model extends the YAGO model by introducing the temporal dimension for facts. The T-YAGO model, adopts all the specification of YAGO model for relations, classes, literals, and statements (facts).

The YAGO model was proven to be very valuable for knowledge exploration and querying. However, it is mostly blind to the temporal dimension of facts. It may store birth dates and death dates of people, but is largely unaware of the temporal properties of events. For example, YAGO may store that a particular person is the prime minister of a particular country, but it does not store the time interval in which this perwon was prime minister. Since, facts change over time, T-YAGO introduces the concept of temporal facts, as explained in Chapter 1. A temporal fact is a relation with an associated *validity time*. It may be valid at a time point or within a time interval.

Although YAGO does not introduce a temporal dimension for facts, its representation model allows us to represent temporal facts via reification. T-YAGO extensively utilizes reification in order to represent temporal facts which are higher-order facts.

Higher-order fact: A higher order fact is a statement which contains a fact as the subject of the statement. Informally, higher order facts are statements about statements. In order to support the binary relation model of YAGO, T-YAGO decomposes a higher order fact into a *base fact* and several associated facts temporally qualifying the base fact. The T-YAGO model assigns a fact identifer to a base fact, and then the associated facts are represented as the relation between the identifier and the remaining arguments. For example, the fact "*Abdullah Gül was prime minister of Turkey from 2002 to 2003.*" is represented in T-YAGO as follows:

- `#14` `Abdullah_Gül` *holdsPoliticalPosition* `Prime_Minister_of_Turkey`
- `#15` `#14` *startedOnDate* ‘‘*2002*’’
- `#16` `#14` *endedOnDate* ‘‘*2003*’’

where the fact `#14` `Abdullah_Gül` *holdsPoliticalPosition* `Prime_Minister_of_Turkey` is the base fact and the others are associated facts temporally qualifying the base fact.

For higher-order temporal facts T-YAGO uses three temporal relations to show the validity time of the facts; *startedOnDate, endedOnDate, happenedOndate*. For the temporal facts that are valid during a time period, T-YAGO uses *startedOnDate* to represent

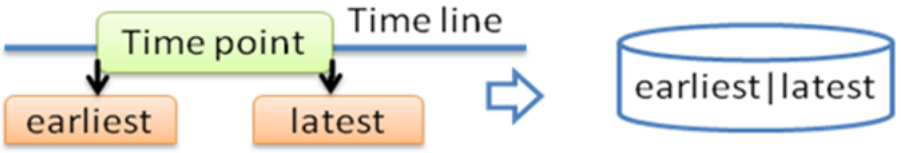

FIGURE 3.3: T-YAGO date data type.

the starting point of a time interval and uses *endedOnDate* for ending point of a time interval. For the temporal facts valid only at a time point, T-YAGO uses the relation *happenedOnDate*.

Apart from higher-order temporal facts, T-YAGO also introduces the notion of binary temporal facts, in order to avoid the computational complexity of reification. Whenever, a temporal fact can be represented in a binary relation, T-YAGO prefers this. For example, the temporal fact "*Max Planck Gesellschaft was founded on 1948*" can be represented without reification as:

- `#17 Max_Planck_Gesellschaft` *wasEstablishedOnDate* `''1948''`

Sometimes it is impossible to extract accurate time points, or sometimes T-YAGO extractors may only extract the starting or ending point of time of a temporal fact, but not both. For these situations, the T-YAGO model uses the *earliest-latest* data type to extend YAGO's *date* literal.

Earliest-latest data type: T-YAGO introduces the earliest-latest data type for time points with the earliest and latest possible time to constrain the range of the true time point. The granularity of the date literal, that T-YAGO accepts, is an exact day, such as 23-12-1985. T-YAGO extractors use place holders for incomplete data. For example, the fact `#18 Max_Planck_Gesellschaft` *wasEstablishedOnDate* `''1948''` does not obey the T-YAGO date constraints. Thus, the date value `''1948''` is normalized as `''1948-##-##''` via introducing place holders. Then, the fact is stored in T-YAGO knowledge base as `#18 Max_Planck_Gesellschaft` *wasEstablishedOnDate* `[1948-01-01, 1948-12-31]` by determining the earliest and latest bounds of the year 1948[2].

If we later have an evidence that Max Planck Gesellschaft was founded on 26 February 1948, then the particular date in the fact will be refined as `#18 Max_Planck_Gesellschaft`

[2]However, in the rest of this thesis, we will write temporal facts like as `#18 Max_Planck_Gesellschaft` *wasEstablishedOnDate* `1948-##-##` by omitting the boundary dates for the sake of clearity.

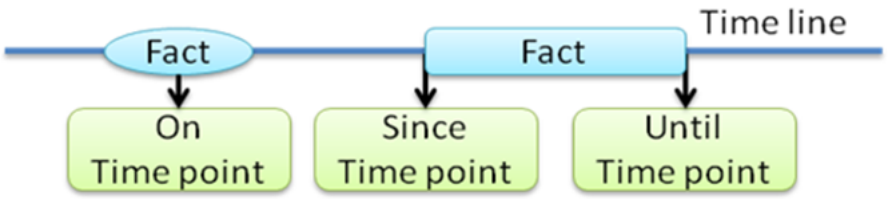

FIGURE 3.4: T-YAGO representation of time points and durations. "*On Time Point, Since Time Point, Until Time Point*" are associated with the relations `happenedOnDate, startedOnDate, endedOnDate`, respectively.

wasEstablishedOnDate `[1948-02-26, 1948-02-26]`. Figures 3.3 and 3.4 summarize how T-YAGO represents validity times of facts.

YAGO2

YAGO2 [HSB+] is an extension of the YAGO knowledge base, in which entities, facts, and events are anchored in both time and space. It is inspired by T-YAGO project to create a temporal knowledge base. Moreover, it also has the location dimension. YAGO2 is built automatically from Wikipedia, GeoNames, and WordNet. It contains 80 million facts about 10 million entities with an accuracy of 95% [HSB+]. YAGO2 has a novel knowledge representation model which is an extension of SPO triple model. The details about YAGO2 go beyond of this thesis.

Since YAGO2 has a vast amount of precise facts and entities, we utilize it during harvesting free text as it is explained in Chapter 5

Chapter 4

Temporal Fact and Event Extraction from Semi-structured Text

The temporal fact extraction from Wikipedia consists of two main phases, extraction from semi-structured text and extraction from free text. The former is accomplished via harvesting infoboxes and categories of Wikipedia articles, and the latter is achieved by harvesting free text of Wikipedia articles. As shown in [ABL+07], infoboxes can be exploited in order to have large amount of precise facts. Moreover, the category pages, which are lists of articles that belong to a particular category, can be exploited to yield high number of facts as it is shown in [Kin05]. In this chapter we investigate methods to harvest infoboxes and categories in a systematic and automatic fashion.

Before diving into the extraction part, we give definitions of temporal facts (simply t-facts) and events in Section 4.1. Then, we will discuss temporal fact and event extraction from infoboxes and categories in Section 4.2 and in Section 4.3.

4.1 Definition of T-fact and Event

Definition: A temporal fact is an RDF triple having a relation which has a temporal value as object. For example the relations such as *wasCreatedOnDate, wasBornOnDate, publishedOnDate, etc.* have temporal value as object of the relation.

Example facts:

- **f1:** `Bob_Marley` *wasBornOnDate* `06-02-1945`.
- **f2:** `A_Beautiful_Mind_(book)` *publishedOnDate* `1998-##-##`.
- **f3:** `Hagia_Sophia` *wasCreatedOnDate* `537-##-##`.

The relations *hasWonPrize, playsForTeam, holdsPoliticalPosition, isMarriedTo, etc.* do not have temporal information as objects. However, those relations can be reified with direct temporal relations such as *happenedOnDate, occursSince, occursUntil*.

Example facts:

- **f11:** `Arnold_Schwarzenegger` *holdsPoliticalPosition* `Governor_of_California`.
- **f12:** `f11` *occursSince* `17-11-2003`.
- **f13:** `f11` *occursUntil* `03-01-2011`.
- **f33:** `Orhan_Pamuk` *hasWonPrize* `Nobel_Prize_in_Literature`.
- **f34:** `f33` *happenedOnDate* `07-12-2006`.

Although there is no universal definiton of *event*, some examplary definitions can be given, such as Wordnet which has three different definitions of event:

- something that happens at a given place and time
- a special set of circumstances.
- a phenomenon located at a single point in space-time.

According to the Cambridge English Dictionary, an event is *"anything that happens, especially something important or unusual"*. In philosophy, events are objects in time or instantiations of properties in objects. However, a definite definition has not been reached, as multiple theories exist concerning events.

In our case we define an event as follows:

Definition: An event is a semantic concept which implies a certain starting or ending time point of a temporal fact. An event is not explicitly encoded in our knowledge base. However, we mention it as a semantic notion.

For example the fact `Bob_Marley` *wasBornOnDate* `06-02-1945` implies the *Birth_Of_Bob_Marley* which is an event. We take such implications into account during event detection from free text. We convert some temporal facts into its semantic event, since events frequently occur in adverbial phrases, such as "*after birth of Frank Zappa, before presidency of Obama, etc.*". (See Chapter 5 for more details)

There is also notion of named event.

Definition: A named event is an entity which has a certain time period or time point directly attached to itself, such as *French Revolution, World War II, Treaty of Lausanne, FIFA World Cup 2006, 37th G8 summit, 23rd Grammy Awards, etc.* In general, named events are important historical events, such as wars, conferences, sport events, treaties, etc. Named events are explicitly encoded in T-YAGO, and their type is `T-YAGOEvent`. We focus on named events during infobox harvesting, since most of the articles talking about named events have infoboxes.

4.2 Extraction from Infoboxes

As many systems such as YAGO [SKW07b] and DBpedia [ABK+07] use infoboxes as the most valuable source of information in Wikipedia articles, we also exploited infoboxes in the framework of temporal fact extraction. Wikipedia infoboxes are essentially typed records of attribute-value pairs. Infoboxes usually contain the most significant information about the entity described by the article. For example, the infobox of the Wikipedia article about French Revolution, as in Table 4.1, contains information about the name of the event, its location, its date, the participants of the event, and more.

A major challenge in harvesting infoboxes is the structural diversity that infoboxes have. Every infobox object has a related Wikipedia template which renders it into HTML format. During the evolution of Wikipedia, a large variety of templates has been used to represent the same or similar information. This results in different types of infoboxes with different attributes and different units of measurements for representing equivalent information. There are about 2,500 distinct infobox types and each of them has about 20 pairs of attribute and value on average [BC10]. However, there is a long tail in the distribution of the number of occurrences of the infobox types as it is shown in [BC10]. Thus, it is plausable to take the most popular infobox types into consideration to generate a set of rules for extraction. In this section, we focus on infoboxes of named events such as *French Revolution, Turkish War of Independence, 2010 FIFA World Cup, French legislative election, 2007, etc.* Our aim is to extract as many named events as possible from infoboxes. Each infobox has a particular type, such as *Historical event,*

```
{{Infobox Historical event
| Event_Name = The French Revolution
| Image_Name = Prise de la Bastille.jpg
| Image_Caption = The storming of the Bastille, 14 July 1789
| Participants = French society
| Location = [[France]]
| Date_start = 1789
| Date_end = 1799
| Result = [[Proclamation of the abolition of the monarchy]]
}}
The French Revolution began in 1789 with the convocation of
the [[Estates-General of 1789| Estates-General]] in May.
The first year of the Revolution witnessed members of the [[Third Estate]]
proclaiming the [[Tennis Court Oath]] in June,
the [[Storming of the Bastille | assault on the Bastille]] in July, the passage
of the [[Declaration of the Rights of Man and of the Citizen]] in August, and
an [[The Women's March on Versailles|epic march]] on [[Versailles]] that forced
the royal court back to [[Paris]] in October.
etc.
[[Category:French Revolution]]
[[Category:18th-century rebellions]]
[[Category:18th-century revolutions]]
```

TABLE 4.1: Wikipedia Markup Language

Election, Military conflict, Competition, etc. For example, the infobox in table 4.1 has type *Historical event*. These infobox types are useful in order to understand whether an entity is a named event. In other words, they will be used to construct a particular fact about the entity via `type` relation. The infoboxes are harvested by the help of an attribute map as it is explained in [Suc]. A complete list of named event infoboxes are shown in Table 6.2.

Definition: Attribute Map

An attribute map is a function which maps an infobox attribute to a pre-defined relation.

An *attribute map* takes several attributes, such as, date, year_started, term_start, etc. and maps each of them to a target relation, such as `happenedOnDate, occursSince, occursUntil`. Each relation has a specific domain and range. One attribute maps to one and only one relation, whereas a relation might have many attributes mapped to it. The date value of a particular attribute has to be normalized in order to be consistent with the T-YAGO data model. Moreover, the type of the infobox is checked to identify whether the entity is a named event. Then, the system will connect the entity to T-YAGO literal `TYAGOEvent` via the relation `type`. Thus, each named event entity will

have the type `TYAGOEvent`.Example event facts that can be extracted from an infobox would be as follows:

- **f1:** `French_Revolution` *type* `TYAGOEvent`.
- **f2:** `f1` *occursSince* `1789-##-##`.
- **f3:** `f1` *occursUntil* `1799-##-##`.

Algorithm 1 Harvesting Infoboxes for Named Events

procedure HARVEST_EVENTS(Wikipedia corpus W)
  $F \leftarrow \emptyset$, ▷ Facts about named events
  create set of infobox types T for named events
  create attribute map A
  for all $w \in W$ **do** ▷ for all articles w
    if $w.hasInfobox()\&(w.getInfoboxType() \in T)$ **then**
      $entity \leftarrow createEntity(w)$, ▷ create the entity associated to article w
      $fact \leftarrow$ CREATEFACT$(w, type, TYAGOEvent)$, ▷ create the event fact
      $F \leftarrow F \cup fact$, ▷ Store fact
      for all $attribute_i \in w.infobox$ **do** ▷ for all attributes in infobox of w
        if $A.contains(attribute_i)$ **then**
          $relation \leftarrow A.getTargetRelation(attribute_i)$ ▷ The target relation is determined
          $value \leftarrow parseValueOf(attribute_i)$ ▷ Value of $attribute_i$ is parsed
        if *value* is in range of *relation* **then**
          $F \leftarrow F \cup createFact(fact, relation, value)$ ▷ The event fact is reified with *value* and by *relation*

The process for harvesting event facts from infoboxes is shown in Algorithm 1. The algorithm traverses all Wikiedia articles. If it detects an infobox, it checks whether the infobox is in the list of named event infoboxes. If so, then the algorithm creates the type fact showing that the entity is a named event. Then, it goes through all attributes of the infobox. For each attribute, it looks up the manually created attribute map to check whether the attribute is listed in the map [1]. If the attribute map contains the attribute, the algorithm parses the value of the attribute and checks whether it is in the range of the target relation of the attribute. If the value of the attribute is in the range of the relation, then the event fact is reified by the target relation and value of the attribute, which results in another fact. For example, if the infobox in 4.1 is given to the algorithm, then it will create the event fact

f1: `French_Revolution` *type* `TYAGOEvent`.

by detecting infobox type. Next, the attribute `Date_start` will be mapped to relation `occursSince` which has the range `yagoDate`.And a regular expression based date parser is used to parse the value of attribute `Date_start` to get date `1789-##-##` which is in

[1] See Table 6.2 for list of attributes

Infobox Type	**Attribute having the temporal information**
Military Conflict	date
Election	election_date
Treaty	date_signed
Olympic event	dates
Historical Event	Date
Film	released
Football match	date
UN resolution	date
MMA event	date
Wrestling event	date
Music festival	dates
Awards	year
Attack	date
Accident	dates
Swimming event	dates
Competition	year
Congress	start|end
Tournament	fromdate|todate

TABLE 4.2: Event infobox types and attributes. An attribute map contains all the attributes of event infoboxes.

the range of `yagoDate` . Here we use the date parser from [SIW06]. Then, the event fact **f1** is reified with normalized date to create new temporal fact as below.

f2: `f1` *occursSince* `1789-##-##`.

4.3 Extraction from Categories

Each Wikipedia article contains a part called *categories*. All categories form a hierarchical structure called **Wikipedia Category Graph**. Each category may have many subcategories and articles. A subcategory is created due to a *hyphonymy* or *meronymy* relation. To illustrate, a category named *Civil rights movement during the Lyndon B. Johnson Administration* is a subcategory of *Civil rights movement*. Each article may link to a number of semantically related categories. For example, the article named *Luso-Chinese agreement* is under the category *Treaties of the Kingdom of Portugal* and also under the category *History of Macau*.

In order to gather information from categories of Wikipedia, one could select one or more top nodes which best describe the information required and then iteratively fetch the subcategories. However, this approach will fail due to the fact that Wikipedia does

not impose a crips hierarchical structure[ZG07]. Furthermore, it may have disconnected or dangling categories, too. Besides the challenges mentioned so far, Wikipedia does not convey a clear classification of categories. Some categories may occur under unrelated categories. For instance, the category *Low-carbon economy* and the category *Vehicles by fuel* are under the same category named *Electric vehicles*.

Among more than 500 thousand categories of Wikipedia, about 70 thousand contain temporal information, such as *Conflicts in 2008, 1997 in international relations, Candidates for the French presedential election 2007, etc.* (See Figure 3.1.) The problems mentioned earlier are also valid for temporal categories of Wikipedia. For example, the article *23rd G8 Summit* is under the category *1997 in the United States, 20th Century Conferences* and also under the category *History of Denver-Colorado*. Therefore, it is not easy to establish a canonical form of temporal expressions appearing in categories for harvesting. Moreover, the syntactic structure of categories shows many variations. Although, many categories include temporal information, they hide this temporal information in many forms. This difficulty pushed us to investigate and analyze categories of Wikipedia so as to declare efficient rules for harvesting temporal categories of Wikipedia.

2010 data dump of Wikipedia contains more than 500K categories. Among those categories, around 70K categories are heuristically extracted as having temporal information. The heuristics employed here are types of regular expressions which check each category whether it has a month, year, or century pattern. The accuracy of these heuristics are shown in Chapter 6. An analysis of temporal categories showed us that an important fraction of temporal categories follows some implicit semantic patterns, which allow us to construct rules in order to extract important temporal facts and events. The analysis yielded 13 conceptual categories which have important events and are clean enough. To illustrate, the category *1834 elections in the United States* has only the elections held on 1834 such as, *New York gubernatorial election*. And the name of the category shows that it includes named events, basically elections held on 1834. However, the temporal category *History of Poland (1945-1989)* contains important people of that era, battles, disasters, and so on. Therefore, the temporal category *History of Poland (1945-1989)* does not contain a certain type of articles, which causes noise during extraction. For this reason, we developed rules only focus on temporal categories for which we can develop canonical forms to create temporal and event facts. These temporal categories are divided into 13 conceptual classes as below:

- Politicial events, such as elections, battles, conflicts, etc
- Establishments, such as establishment of important instutions, parties, etc.
- Disestablishments, such as disestablishment of important instutions, parties, etc.

FIGURE 4.1: Categories belonging Nicolas Sarkozy's Wikipedia article

- Discoveries, such as geographycal places, inventions, etc.
- Disasters, such as earthquakes, fires, floods etc
- Accidents, such as transportation accidents, submarine accidents, etc.
- Architectural events, such as construction of important buildings.
- Laws, such as important orders, acts, resolutions, etc.
- Important releases, such as music albums, movies, etc.
- Important publications, such as books, novels, etc.
- Artistic works, such as, operas, musicals, paintings, etc.
- Births
- Deaths

These conceptual categories are employed for extraction of events and temporal facts. Connecting the temporal information extracted from a temporal category to an entity belonging to that category requires some elegancy. It is obvious that *23rd G8 Summit, Turkish War of Independence* are named events, whereas entity *Michael Jackson*, which appears under category *2009 deaths*, is neither a named event nor a temporal fact. However, we can combine the conceptual class of the category and the entity to determine the relation `diedOnDate`, and we can create the fact `Michael_Jackson` *diedOnDate* `2009-##-#-#`. Here we employ a *Category-relation map* to extract the proper relations from categories.

Definition: Category-relation map

*A **Category-relation map** is a function which maps a conceptual class to corresponding pre-defined relation by looking at the keywords occurring in the category name.* To

illustrate, the temporal category *Vehicles introduced in 1999* is first mapped to *Discoveries* conceptual class, since the category *Vehicles introduced in 1999* has the keyword *introduced*. Then, *Category-relation map* gets the target relation `wasDiscoveredOnDate` for the conceptual class *Discoveries*. For a detailed mapping of conceptual classes and relations, please see Table 4.3.

After applying Category-relation map to a category, we construct some example temporal facts as below:

- **f1:** `Michael_Jackson` *diedOnDate* `2009-##-##`.
- **f2:** `Audi_S3` *wasCreatedOnDate* `1999-##-##`.

Another approach is to combine temporal information directly to the entity via a proper relation without reification, as below.

- **f12:** `Michael_Jackson` *diedOnDate* `2009-##-##` .
- **f13:** `Audi_S3`*wasIntroducedOnDate* `1999-##-##` .

Harvesting Categories

Temporal fact and event extraction from categories can be done in two ways: First, extracting all categories of Wikipedia and traversing them to create temporal facts. Second, traversing all articles and looking at categories of them to construct temporal facts. The first approach has the advantage of modularity; it can be run anytime to extract facts from categories without parsing the entire article dump of Wikipedia. In contrast, the second approach has to be merged into the whole extraction procedure, extraction from infobox and categories. That is the reason why YAGO chose the second approach. YAGO uses declared rules to extract temporal facts from categories of the article at hand. However, in the first approach, the articles do not have to be processsed, which makes it faster. Here, we will explain the first approach.

First of all, temporal categories of Wikipedia are extracted by simple heuristics which are around 70K out of 500K categories. By employing semantic word similarity, the temporal categories are classified into 13 conceptual classes which are explained above. The temporal information that is used as time point of facts are extracted from categories by employing a regular expression based date extractor and normalizer. For each temporal category, the entities belonging to it are fetched and processed by Category-relation map to determine the target relation. Finally, the facts are constructed via joining temporal

Conceptual Class of Events	**Keywords**	**Target Relation**
Political events	election, war, battle	happenedOnDate
Disasters	disaster	happenedOnDate
Events related to legislations	law	happenedOnDate
Establishments	establishment, commission, open	wasCreatedOnDate
Disestablishments	disestablishment, scuttle, close	wasDestroyedOnDate
Discoveries	discovery, description, notification, introduction	wasDiscoveredOnDate
Accidents	accident, incident	happenedOnDate
Architectural Events	construction	wasCreatedOnDate
Important releases	release	wasCreatedOnDate
Important publications	publication	wasCreatedOnDate
Artistic works	Work	wasCreatedOnDate
Births	birth	wasBornOnDate
Deaths	death	diedOnDate

TABLE 4.3: Category-relation map

information provided by the particular category and the entity. The overall process is depicted via a flowchart in Figure 4.2, where rectangular nodes represent the particular data and edges represent the data flow. The diamond node represents Category-relation map, and the sylyndirical node represents our knowledge base T-YAGO.

4.4 Conclusion

In this chapter, we have presented the difference and similarity between events and temporal facts. Then we explored methods to harvest infoboxes and categories of Wikipedia articles. Since, infoboxes and categories supply us semi-structured data, we used rule based methods to extract temporal facts and events from infoboxes and categories. The main focus of infobox harvesting is named events. However, most of the Wikipedia articles do not have infoboxes, which causes low recall. To come over the challenge of low recall, we introduced techniques about how to extract more temporal facts and

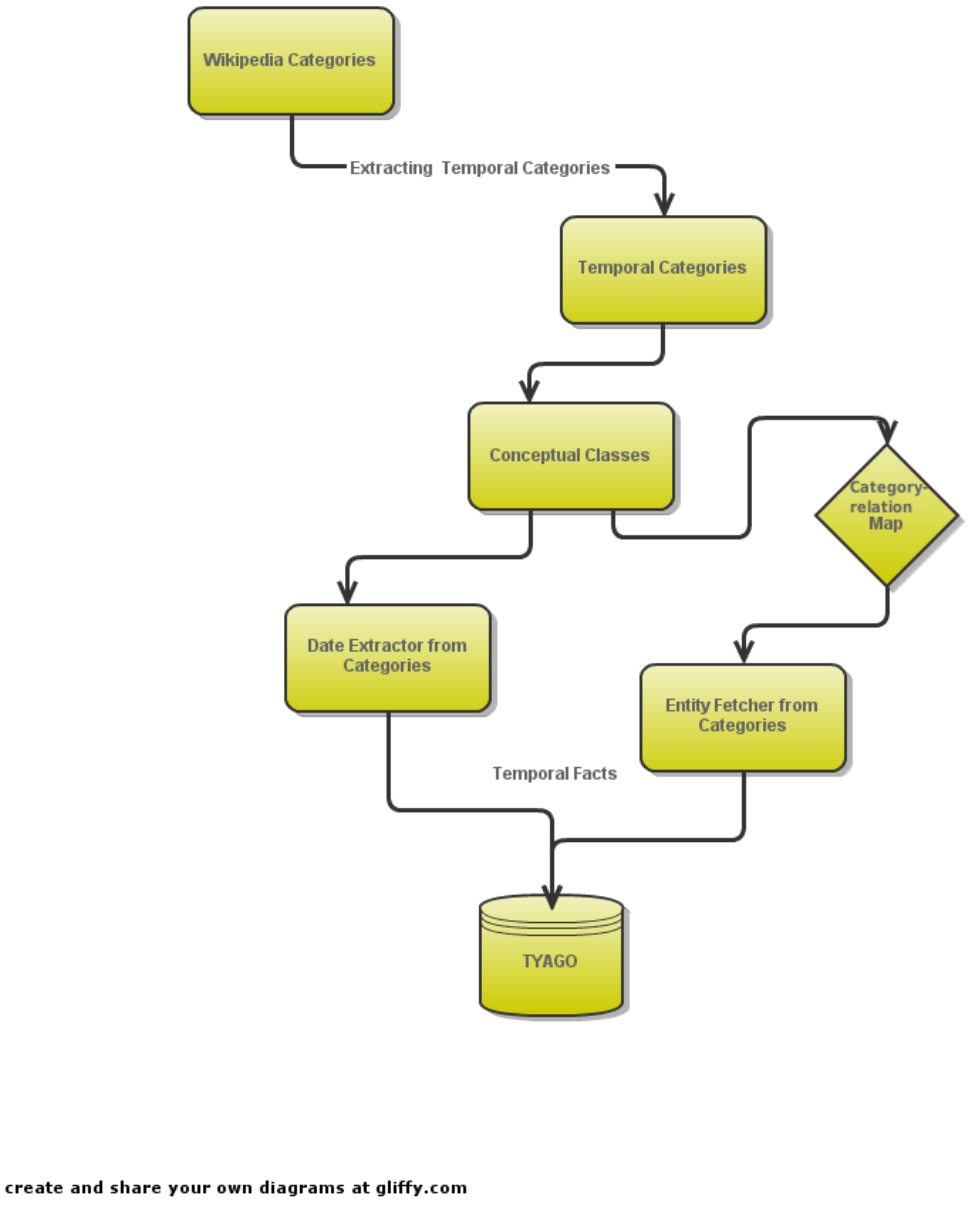

FIGURE 4.2: The process of harvesting temporal facts and events from Wikipedia categories

events from Wikipedia categories. We gathered large amount of facts from Wikipedia categories which are shown in Experiments section in details.

Chapter 5

Temporal Fact and Event Extraction from Free Text

In the previous chapter it is shown that categories and infoboxes of Wikipedia articles can be harvested by declarative IE approaches, which yields highly precise temporal facts. However, since the high quality semi-structured data is often limited and most web sources are unstructured, it is necessary to extract temporal facts from arbitrary free texts (in our case the text source of Wikipedia articles).

The 2007 TempEval Challenge [VGS+07] is an established workshop in the field of temporal information extraction from free text. It focuses on tasks which involve identifying event-time and event-event temporal relations. In TempEval Challenge only a restricted set of Allen-style [All83] temporal relations are used. The state-of-the-art system in the field of temporal information extraction is TARSQI [VP08] which succeeded in detecting temporal expressions by marking events and generating temporal associations between times and events. Another system called TIE [LW10] did not only detect the temporal relationships between times and events, but also used probabilistic inference so as to bound the time points of the beginning and the end of each event. It is important to note that the concept of *event* in the above systems or tasks is different from our concept of *fact* or *event*. Rather than checking relationships of base facts, both systems focused on extracting time information either by time-events or by event-events. Here, events are defined as verb forms with tenses, adjectives, and nouns that describe the temporal aspects of the events. In the example of "Obama accepted the award in Oslo last year.", systems like TARSQI consider the verb "accepted" to be an event, and adverbial phrase "last year" to be a time point. This task however is very different from our notion of events and temporal facts referring to entities and relations, as defined in Chapter 4.

In this chapter, we present techniques to extract temporal facts and events from the free text of Wikipedia articles. We first introduce pattern-based extraction by normalizing explicit temporal expressions. Next, we discuss dictionary-based extraction by normalizing implicit time expressions.

5.1 Pattern-based Extraction by Normalizing explicit Dates

In this section we present how to exploit T-YAGO for its own growth by using trusted facts as the basis for generating good text patterns. Applying those patterns on free text will generate new temporal facts. Extracting new facts by patterns has challenges regarding *correct pattern selection, entity disambiguation, coreference resolution* and *type checking*, respectively.

Pattern Selection: Temporal facts are extracted from free text by finding meaningful patterns in the text. The precision of this technique strictly depends on having a variety of meaningful patterns. Thus, discovering and assessing patterns is a significant task of IE.

We propose a pattern generation algorithm which benefits from the knowledge base. Moreover, the frequency of patterns are statistically assessed in order to have frequent patterns which will result in high recall and high precision.

Entity Disambiguation: For ontology oriented information extraction, the phrases from text have to be mapped to correct entities in the ontology. In most of the cases, this mapping is ambigous, which makes finding the intended meaning of a phrase difficult. For example, the word *"Andrew"* may map to *Andrew Jackson* or *Andrew Johnson*.

We use heuristics to disambiguate entities. First, our system can understand the current Wikipedia article (recall that each article maps to one only one entity) which is being processed. Then it determines possible words that can map to entity by tokenizing the entity. For example, if the entity is *Michael Jackson*, then the word *Michael* or the word *Jackson* is mapped to the entity *Michael Jackson*. However, if both words occur adjacent, then it is directly mapped to the entity.

Coreference Resolution: In natural language documents, there are multiple expressions in a sentence or document referring to the same thing. For example, in "John is married to Mary. He loves her so much." The word *"He"* refers to *"John"*. In order to derive the correct interpretation of text, or even to estimate the relative

importance of various mentioned subjects, pronouns and other referring expressions need to be connected to the right individuals. Although there are tools for anaphora resolution such as [PK04, VPP+08, QKC04], they have complex algorithms which causes degradation in performance. Moreover, the results have a typical precision less than 70 %.

We use basic pronoun heuristics to overcome the problem of coreference resolution. Since our system can determine the current article (entity) at hand, we replace the pronouns such as *his, her, he, she, him* with the name of the entity. It is intuitive, since in most of the cases a Wikipedia article uses pronouns referring to the entity itself. This pronoun heuristics, surprisingly works with a fair level of precision (See experiments and evaluation section).

Type Checking: An ontology is not a simple collection of facts, but it is a collection of typed entities and facts creating a taxonomical structure. The newly extracted facts have to be consistent with the domain and range of the relations. To illustrate, while the fact **f1**: *Nicolas_Sarkozy* `isMayorOf` *Neuilly-sur-Seine* is correctly typed, i.e., the entities are in the domain and range of the particular relation, the fact **f2**: *Nicolas_Sarkozy* `isMayorOf` *French_Economy* is not correctly typed because *French Economy* is not in the range of `isMayorOf` relation. Therefore, **f2** has to be pruned.

In order to overcome the challenge of type checking, we keep a list of entities that are in the range of the given relation. If the newly extracted fact has an argument which is not in the range, then the fact is pruned.

Although there are pattern-based systems such as SOFIE [SSW09] and PROSPERA [NTW11] to extract RDF style facts from free text, they only focus on extracting base facts rather than extracting higher-order temporal facts. Extraction of higher-order temporal facts is more difficult, since temporal facts have to be reified with correct temporal value qualifying the base fact. To illustrate, from the sentence "*Jacques Chirac, born on 29 November 1932, is a French Politician who served as President of France from 1995 to 2007 and served as Prime Minister of France between the years 1974-1976 and from 1986 to 1988.*" **f1:** `Jacques_Chirac` *holdsPoliticalPositon* `President_of_France` is a base fact and can be extracted via human engineered rules which will search for a certain pattern. However, if we want to extract the time interval for the fact **f1**, then we have to determine the correct date qualifying **f1** and connect it to **f1** as **f2:** `f1`*startedOnDate* `17-05-1995`,**f3:** `f1`*endedOnDate* `16-05-2007`.

The temporal fact extraction has 2 phases:

1. **Phase 1:** Detection and normalization of temporal expressions
2. **Phase 2:** Extraction of base facts and reifying the base facts with correct temporal information

For the first phase, we use a regular expression based date parser [SIW06] to parse and normalize large variety of temporal expressions. For the second phase, we suggest to use a pattern-based extraction. We automatically create patterns which have place holders for base facts and also for temporal information qualifying the base fact. Thus, instances of such patterns can be used to create base facts and refied fact. As an example, the pattern <*politician*> *served as* <*political office*> *from* <*date*> *to* <*date*> will extract the fact **f1:** `Jacques_Chirac` *holdsPoliticalPositon* `President_of_France` and reify it as **f2:** `f1` *startedOnDate* `17-05-1995`,**f3:** `f1` *endedOnDate* `16-05-2007` from the sentence "*Jacques Chirac, born on 29 November 1932, is a French Politician who served as President of France from 1995 to 2007 and etc.*"

Our pattern-based temporal fact extraction model consists of two consecutive stages:

1. **Automatic Pattern Induction:** Automatically creating patterns via using the already built temporal knowledge base (T-YAGO)
2. **Application of Patterns:** Applying patterns on free text to create new temporal facts.

Before explaining pattern-based temporal extraction, we give definitions for *seed entity, pattern*, and *pattern matching*.

Definition: A seed entity is an entity manually given to the pattern induction process, in order to learn the possible patterns that creates the known facts about the entity.

Definition: A pattern is a set of string tokens which are defined in a specific template.

Definition: Pattern matching is the act of extraction of string sequences which obeys to a certain template.

1) Automatic Pattern Induction

In this part, our goal is to automatically create patterns for given relations by using T-YAGO. The process is bootstrapped by arbitrary seed entities. For each seed entity, all temporal facts having the seed entity as subject and the particular relation are

fetched. Each temporal fact is decomposed to its arguments, such as *entity_1, entity_2*. Moreover, the temporal interval of facts are retrieved and attached to arguments as *entity_1, entity_2, date_1, date_2* where *date_1* is the time that the particular fact started existing and *date_2* is the time that the particular fact ended existing. We put *entity_1, entity_2, date_1, date_2* in a data structure called ***args***. For each seed entity, the sentences in which those arguments occur together are extracted from Wikipedia articles of the seed entities. The words or phrases between arguments are kept to be used to create a pattern, whereas, the words appearing before the argument *entity_1* and after *date_2* are omitted. All the patterns are put in *patterns vector* which maintains the frequency information of patterns. The process is completed for all seed entities. Finally, only the most frequent patterns are selected to be applied on entire corpus to get new temporal facts.

2) Application of Patterns

Having the frequent patterns in hand, it is possible to get new temporal facts. The patterns are used to find the relation between entities and also the dates. Hence, via using the relation and the entities together, we can create a base fact. Then the base fact will be refified with particular dates obtained by the pattern. As an example, the pattern *<politician> was inaugurated as <political office> on <date>* is created for the relation `holdsPoliticalPosition`. If this pattern is applied to the sentence "*Barack Obama was inaugurated as President of the United States on 20 January 2009.*", then the place holder *<politician>* will be matched to *Barack Obama*, *<political office>* will be matched to *President of the United States*, and the palce holder *<date>* will be matched to *20 January 2009*. Next, the base fact **f23:** *Barack_Obama* `holdsPoliticialPosition` *President_of_the_United_States* is created by using the two entities *Barack Obama, President of the United States* where *Barack Obama* is subject and *President of the United States* is object and the relation is given as `holdsPoliticalPosition`. Next, after normalization of the date *20 January 2009* as *20-01-2009*, the base fact **f23** will be reified as **f24:** *f23* `startedOnDate` *20-01-2009*

A naive explicit match application of patterns will result in a very low recall. To illustrate, if the pattern"*<politicianEntity> also served as <politicalOffice> from <date> until <date>*" is applied to the sentence "Nicolas Sarkozy who is 55 years old served as Minister of the Interior from 2 June 2005 until 26 March 2007", then we will not extract the fact **f1:** *Nicolas_Sarkozy* `holdsPoliticialPosition` *Minister_of_the_Interior* due to seeking for explicit match. However being tolerant to some extent of noise between arguments of the pattern will yield a better recall, yet not sacrificing precision too much. For this reason, we need to change the explicit match function to an approximate

match function which tolerates noise up to a pre-defined threshold. We use word level levenshtein distance as a metric for approximate matching. In previous example, if our tolerance threshold was 5 words, then it would be possible to extract the following facts

f1: *Nicolas_Sarkozy* `holdsPoliticialPosition` *Minister_of_the_Interior*

f2: *f1* `startedOnDate` *02-06-2005*

f3: *f1* `endedOnDate` *26-03-2007*

from the sentence "*Nicolas Sarkozy who is 55 years old served as Minister of the Interior from 2 June 2005 until 26 March 2007*".

The overal procedure of temporal fact extraction is described in Algorithm 2. We bootstrap the extraction by taking a set of seed entities S and temporal relations R in order to induce patterns. All facts belong to seed entities for given relations R are retrieved from the knowledge base K and put in set $F2$. Before inducing patterns, a subcorpus $W2$ is created by retrieving the articles belonging to the seed entities from Wikipedia corpus. (Since $W2$ is much smaller than the entire corpus, the process of inducing patterns will be faster.) Then, each reified temporal fact $f \in F$ is decomposed to its arguments, as like as the quadruple <*entity_1*, *entity_2*, *date_1*, *date_2*> . For example, a fact and its reifications such as, **f1:** *Nicolas_Sarkozy* `holdsPoliticialPosition` *Minister_of_the_Interior*, **f2:** *f1* `startedOnDate` *02-06-2005*, and **f3:** *f1* `endsOnDate` *26-03-2007* is decomposed to the quadraple <*Nicolas_Sarkozy*, *Minister_of_the_Interior*, *02-06-2005*, *26-03-2007*>. All quadruples (or triples in some cases) are stored in a data structure called $args$. Then, each $item_i \in args$, is applied to text and for all satisfying strings of $W2$ new patterns are induced by replacing arguments of $item_i$ with place holders. A small excerpt from the table of patterns for the relation `holdsPoliticalPosition` can be seen in 6.8. Such patterns are able to capture base facts and the dates that qualifying the facts. Moreover, a ranked vector representation of patterns $\mathbf{v}_{(P)}$ is created by using frequency statistics of patterns. Then, for each article $w \in W$, we approximately apply patterns P by using word-based Levenshtein distance to get facts as final output.

5.2 Dictionary-Based Extraction by Normalizing Implicit Dates

As pointed out in the previous section, there is a high amount of temporal information buried in natural language text. To illustrate, although the latest version of YAGO knows the fact *Nicolas_Sarkozy* `holdsPoliticialPosition` *Minister_of_the_Interior*, it does not know the fact *Nicolas_Sarkozy* `holdsPoliticialPosition` *Minister_of_the_Budget*.

Algorithm 2 Temporal Fact Extraction

procedure HARVEST_TEMPORAL_FACTS(seed entities S, corpus W, knowledge base KB, relation R)
  $P \leftarrow$ INDUCE_PATTERNS(S, W, KB, R) ▷ **collect** large numbers of **patterns**
  create and rank pattern vector $\mathbf{v}_{(P)}$ using patterns in P
  $F \leftarrow$ EXTRACT_FACTS$(\{f_i\}, P, M)$ ▷ **collect** large numbers of facts
  return all facts F ▷ facts as final **output**
procedure INDUCE_PATTERNS(seed entities S,corpus W, knowledge base KB, relation R)
  $P \leftarrow \emptyset$ ▷ set of patterns
  $F \leftarrow$ GET_FACTS_FROM_KNOWLEDGEBASE(S, KB, R) ▷ **get** facts about seed entities from **knowledge base** for relation R
  $W2 \leftarrow$ CREATE_SUBCORPUS_OF_SEED_ENTITIES(S, W) ▷ **get** articles of seed entities from **entire corpus**
  for all $f \in F$ **do** ▷ for each fact f
    $args \leftarrow args \cup \{(arg_i(f))\}$ ▷ get all arguments, *quadruples* of the fact
  for all $arg_i \in args$ **do** ▷ for each *quadruple*
    $P \leftarrow P \cup \{$CREATEPATTERN$(arg_i, W2)\}$ ▷ for each satisfying strings of $W2$, replace arg_i with place holders
  return patterns P
procedure EXTRACT_FACTS(corpus W, patterns vector V)
  $F \leftarrow \emptyset$ ▷ set of new facts
  for all $w \in W$ **do** ▷ for all articles in W
    for all $p \in V$ **do** ▷ for all patterns
      for all $s \in W$ **do** ▷ retrieve strings s in w approximately matching the pattern p
        $F \leftarrow F \cup \{($CREATE_FACT$(w, p, s))\}$ ▷ creating fact about associated entity of w by applying p on s
  return F ▷ facts created by approximately matching ranked patterns P on W

Automatically Generated Patterns
<politician> was <political office> from <date> to <date>
<politician> served as <political office> from <date> to <date>
<politician> was the <political office> serving from <date> until <date>
<politician> was inaugurated as <political office> on <date>
<politician> notably served as <political office> from <date> to <date>
<politician> was appointed <political office> on <date>
<politician> was elected <political office> in <date>
<politician> was sworn in as <political office> on <date>
<politician> is a politician who was the <political office> from <date> to <date>

TABLE 5.1: Patterns for the relation `holdsPoliticalPosition`

However, there exists an interesting sentence in the Wikipedia article about Sarkozy, "*Nicolas Sarkozy was Minister of the Budget in the government of Edouard Balladur, during Francois Mitterrand's last term.*" which shows that the fact *Nicolas_Sarkozy* `holdsPoliticalPosition` *Minister_of_the_Budget* can be created with valid time intervals once the implicit temporal phrase "*in the government of Edouard Balladur*" or "*during Francois Mitterrand's last term*" are detected and normalized correctly. Since T-YAGO knows the facts

- **f214:** *Edouard_Balladur* `holdsPoliticialPosition` *Prime_Minister_of_France*
- **f215:** *f214* `startedOnDate` *29-03-1993*
- **f216:** *f214* `endedOnDate` *10-05-1995*

our proposal is to employ T-YAGO in order to resolve dates of such implicit expressions. We analyzed Wikipedia articles and found that implicit time phrases frequently appear in adverbial phrases. Therefore, we focus on normalization of temporal adverbial phrases.

In this section our focus is to output a set of matchings M from implicit temporal phrases P to temporal facts which exist in knowledge base. We call this process as *Dictionary Based Event Detection and Extraction*. The remainder of the section is organized as follows. Part 5.2.1 explains how to map temporal phrases to already existing temporal facts. And in part 5.2.2, we present how to use those mappings to give a temporal dimension to the existing base facts.

5.2.1 Mapping Temporal Phrases to T-facts

As it is mentioned in chapter 2, there are three types of temporal expressions, explicit date expressions, relative time expressions and implicit time expressions. In order to extract temporal facts from free text with a high precision and recall, all kinds of temporal expressions have to be normalized and connected to facts. Although our fact extractors can succesfully detect and annotate explicit date expressions (1998, 23rd January 1976, etc.) via our regular expression based date normalizer and relative time expressions (last year, next week, etc.) can be normalized by TARSQI, detection and normalization of temporal adverbial phrases are untouched so far. There is research related to detection and normalization of temporal adverbial phrases [KM10, PV09, LW10, VMS+05], however, they focus on the temporal clauses such as *a year ago, two weeks later, etc.* They ignore the adverbial phrases which implicitly contains temporal facts such as *during Chirac's presidency, before Second World War, after his last term, etc.* Since relative time expressions are not suitable for encyclopedic content, Wikipedia articles do not contain relative time expressions. However, explicit time expressions and temporal adverbial phrases are extensively used, which forced us to investigate methods for normalization of temporal adverbial phrases.

Concisely, given a phrase, we aim to find what the most likely target is for mapping the phrase to a temporal fact which represents that phrase. This mapping process is done in three steps, **identification**, **interpretation** and **normalization** respectively. Each step is explained as follows.

Table 5.2: Patterns for candidate event detection

Pattern
<during>, <Noun Phrase>
<after>, <Noun Phrase>
<before>, <Noun Phrase>

5.2.1.1 Identification of Temporal Expressions

In order to have a precise mapping from implicit temporal phrases to t-facts, we need to identify good candidate phrases from text. For this reason, we bootstrap the identification step starting out with a set of manually defined patterns to detect candidate temporal clauses. After a short analysis of Wikipedia articles, we find out that temporal adverbial phrases usually have the "*<adverbial keyword>, <Noun Phrase>*" structure. Hence, a pattern in the form of "*<adverbial keyword>, <Noun Phrase>*" will find all noun phrases occuring in a temporal adverbial phrase. An instance of such a pattern would be similar to "*<during>,<Noun Phrase>*" which will find candidate phrases, such as, "*during, French Revolution*", "*during, Mitterrand's presidency*", etc.

We manually defined a set of patterns listed in Table 5.2. Each pattern has two crucial parts, a **temporal keyword** such as *during, after, before* and a **noun phrase** part.

Applying patterns requires identifying both parts of the patterns in the text. Although identifying the adverbial keyword is easy, identifying the noun phrase occuring in the adverbial phrase is more challenging. Here, we use a dependency parser in order to get the complete noun phrase occuring in the adverbial clause. A dependency parser assigns a given sentence to a linkage (a syntactic structure), which consists of a set of labeled links connecting pairs of words based on link grammar. For example for the sentence "*After his re-election as President of the French Republic, Chirac appointed Sarkozy as French Minister of the Interior.*", a dependency parser will yield the linkage as in the figure 5.1. A constituent tree is generated from the linkage output as in figure 5.2. The constituent tree shows the noun phrases in NP tags, such as *(NP his re-election)*, *(NP (NP the French) Republic)*, etc. Therefore, by taking the noun phrases occuring in adverbial clause, the temporal phrase "***After his re-election***" is identified by instantiating the pattern "*<adverbial keyword>, <Noun Phrase>*" on the sentence "*After his re-election as President of the French Republic, Chirac appointed Sarkozy as French Minister of the Interior.*".

We use the state of the art tool The Link Grammar Parser [ST95]. The Link Grammar is a robust system which covers almost all aspects of English grammar. It uses a dictionary of English words which has around 60,000 words. Although it is a dictionary based

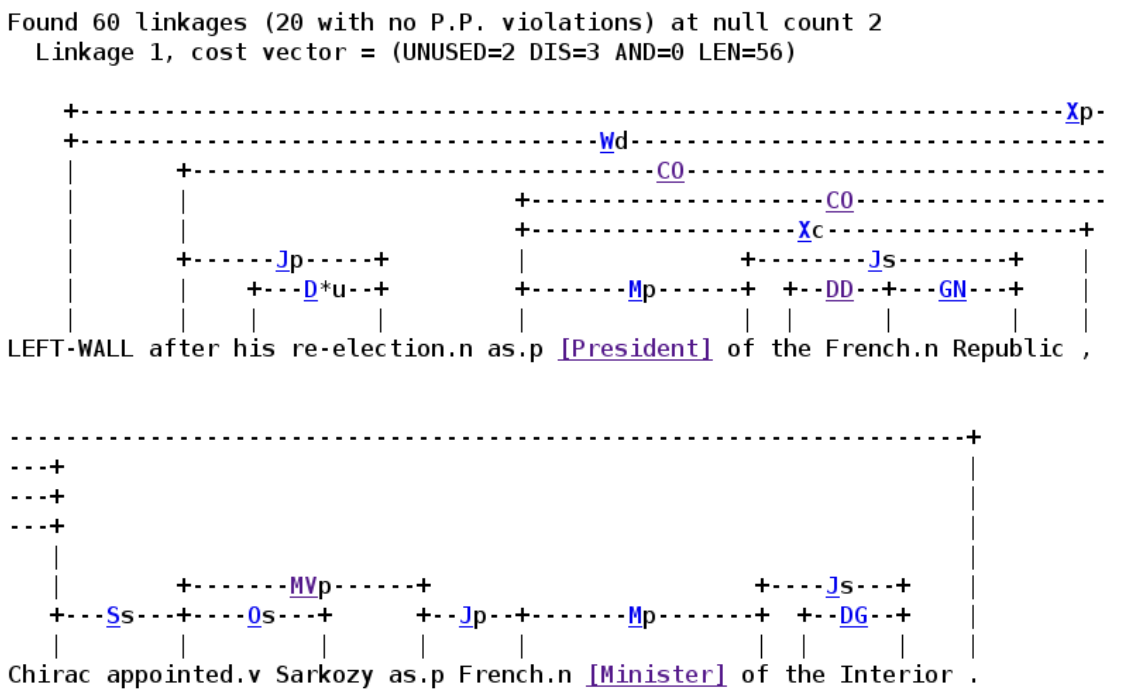

FIGURE 5.1: The linkage found for the sentence "*After his re-election as President of the French Republic, Chirac appointed Sarkozy as French Minister of the Interior.*" by The Link Grammar Parser

system, it can handle sentences impressively well even if they have one or two words which are not in the dictionary. Moreover, it can predict the part-of-speech for these words with a fair degree of accuracy.

As a summary, by using pre-defined patterns and running dependency parser, the identification stage outputs the candidate phrases which will be used as the input of **interpretation** stage.

5.2.1.2 Interpretation

Here we aim to map a given candidate temporal phrase to an existing temporal fact in our knowledge base. Formally we define the process of this mapping as follows:

Given a knowledge base K, the purpose of the interpretation stage is to map the candidate temporal phrases found by identification stage to the most similar fact in K.

For this reason, we developed two dictionary based systems, *Rich Dictionary System (RDS)* and *Context Aware System (CAS)*, respectively. Any dictionary d can answer a query q with a confidence score where q is a candidate temporal phrase found by identification stage. Each dictionary uses an approximate matching procedure to return answers to queries.

```
Constituent tree:

(S (PP After
       (NP his re-election))
   (S (PP as President
          (PP of
              (NP (NP the French)
                  Republic)))
      ,
      (S (NP Chirac)
         (VP appointed
             (NP Sarkozy)
             (PP as
                 (NP (NP French)
                     Minister
                     (PP of
                         (NP the Interior)))))))
   .)
```

FIGURE 5.2: The constituent tree produced from the linkage shown in figure 5.1

1) Rich Dictionary System

In this system, we are eager to answer each query[1] by using the entire T-YAGO as a dictionary. We also added more temporal facts from latest version of YAGO after omitting generic entity-time relations, in order to avoid duplications in our dictionary. For example,YAGO has the fact *Frank Zappa* `wasBornOnDate` *21-12-1940* and the fact *Frank Zappa* `startsExistingOnDate` *21-12-1940*. In such cases we avoid second fact since it subsumes the previous one. Furthermore, we turn the fact into a noun phrase so that it is more likely to be seen in an adverbial phrase. To illustrate, we convert the fact *Frank Zappa* `wasBornOnDate` *21-12-1940* to the noun phrase *birth of Frank Zappa*. Whenever we can map a phrase in text to *birth of Frank Zappa*, then we will replace it by the date *21-12-1940*. The convertions of facts to noun phrases are shown in the table 5.2.1.2. This system uses Jaccard similarity over *character q-grams* for measuring the similarity between a particular query and a temporal fact.

Our goal in this baseline system is to return a temporal fact mapping to a query (a candidate phrase), as much as possible.

[1] A query is a temporal candidate phrase issued to our dictionary based systems, so as to map it to an existing temporal fact.

Relation	**Noun form**
wasCreatedOnDate	*creation of, establishment of, commission of, open of, construction of, publication of*
wasDestroyedOnDate	*disestablishment of, scuttle of, close of*
wasDiscoveredOnDate	*discovery of, description of, notification of, introduction of*
wasBornOnDate	*birth of*
wasDiedOnDate	*death of*

TABLE 5.3: Convertions from facts to noun phrases based on the relation

2) Context Aware System

In this system, we have the same system as *RDS* except that we include contextual information into the dictionary. *RDS* uses a dictionary with more than one million temporal facts taken from T-YAGO. However, such a dictionary of facts is large enough to contain a lot of irrelevant facts for a given query, which may cause noise during mapping process. In Context Aware Dictionary, we aim to exploit contextual information so as to decrease the size of the dictionary, yet using more relevant facts. First of all, the system can understand the Wikipedia article being processed. Each Wikipedia article (except redirection pages, talk and template pages) is associated with one and only one entity. Therefore, the system knows the entity which is under process by the time being. CAS takes Wikipedia Hyperlink Graph (WHG) as input in order to detect the first neighbours of the current entity by looking at the outgoing links from the current entity to other entities. (Wikipedia Hyperlink Graph is implicitly encoded in YAGO by the relation `hasInternalWikipediaLinkTo` which shows an article having a link to another one. Hence, WHG can be generated automatically by traversing YAGO entities.) Then it loads all facts about the current entity which is under process and all facts about first neighbours of the entity. Then it builds the dictionary which is much smaller than *RDS*. For example a subgraph of Wikipedia Hyperlink Graph can be seen in figure 5.2.1.2 in which the entity *Nicolas Sarkozy* and its first neighbours are shown. The edges in the graph represent hyperlinks and nodes represent articles (recall that each article is associated witn one and only one entity). If our current article under process is the article of *Nicolas Sarkozy* then **CAS** creates a dictionary of temporal facts by taking all temporal facts about the entity *Nicolas Sarkozy* and all temporal facts about its first neighbours. First neighbours of *Nicolas Sarkozy* are shown in figure 5.2.1.2. When **CAS** finishes processing an article, it clears the dictionary then builds a new dictionary in the same way for the next article. CAS also uses the same similarity metric as RDS for measuring

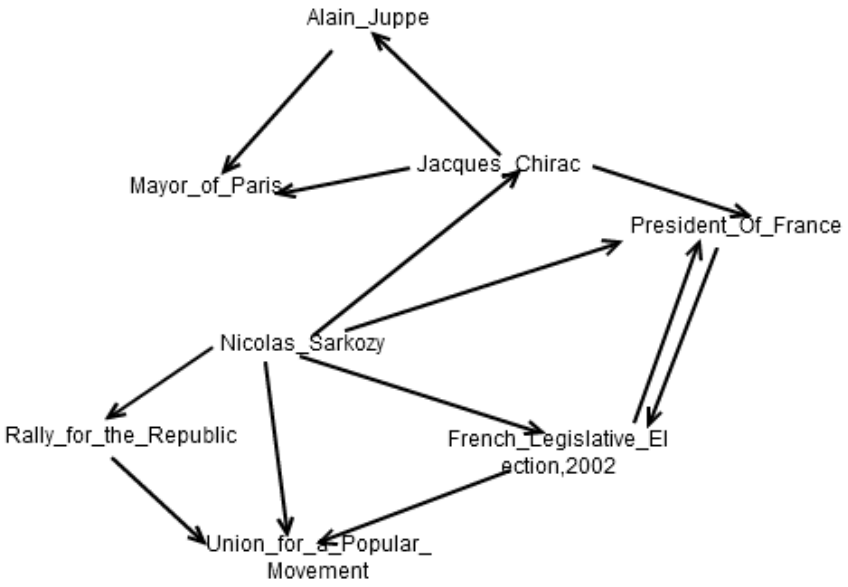

FIGURE 5.3: An excerpt from Wikipedia Hyperlink Graph

the similarity between a particular query and a temporal fact. It also uses the same techniques to convert a temporal fact into a noun phrase which is more possible to occur in an adverbial phrase.

In this system we aim to map a temporal phrase to a temporal fact at the first neighbourhood of the current entity. Since CAS creates a small dictionary for each entity by loading facts related to it, it provides a precise dictionary. However, creating a new dictionary for each entity may cause some degradation in performance.

The interpretation stage outputs temporal which facts are given to normalization stage as input.

5.2.1.3 Normalization

Given a set of temporal facts, the normalization stage ask queries to T-YAGO, in order to get the valid time points or time spans of the temporal facts. Once it gets this temporal information, it annotates the adverbial phrase in text with the temporal information. Thus, the implicit temporal clauses are mapped to time interval or time point that those clauses imply. Normalization stage outputs a temporally annotated text which can be used in section 5.2.2, connecting temporal facts to base facts.

As a summary, in this section we have presented the followings:

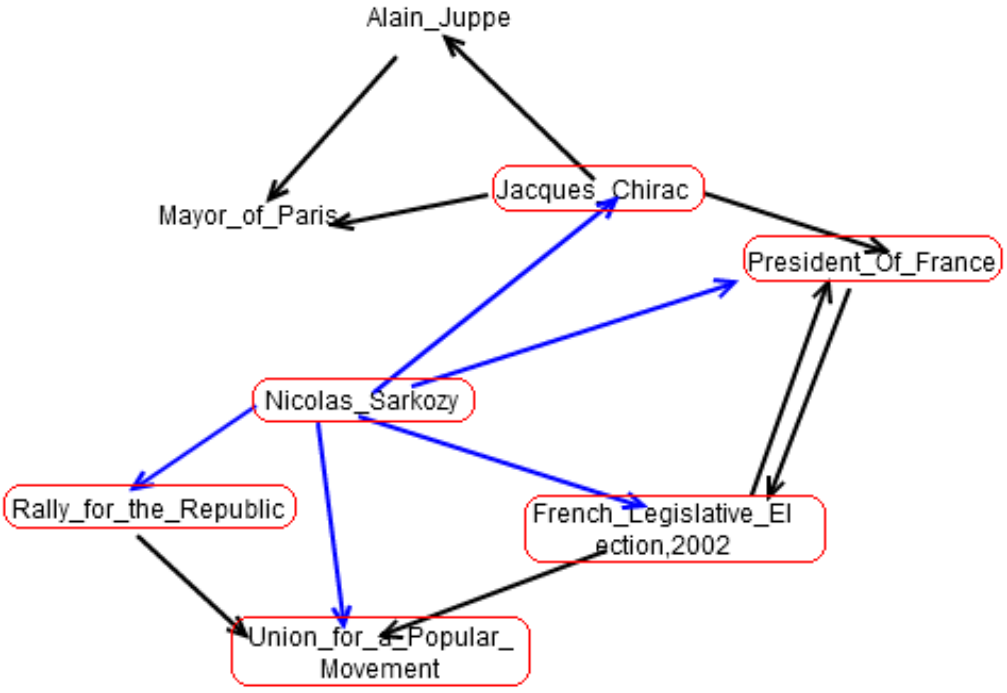

FIGURE 5.4: The graph showing the entity Nicolas_Sarkozy and its first neighbours

- pattern-based identification of temporal phrases
- Exploting the temporal knowledge base to create dictionaries
- An approximate similarity matching to map temporal phrases to temporal facts
- Exploting context (Wikipedia Hyperlink Graph) in order to boost the precision

5.2.2 Connecting T-facts to Known Base Facts

In previous section, we showed how to normalize implicit temporal phrases by using a temporal knowledge base at the background. In other words, we mapped implicit temporal phrases to t-facts. Thus, we have a complete architecture for normalization of temporal expressions apeearing in Wikipedia articles. As a result, we can extract new temporal facts from free text by using the techniques presented in section 5.1. The techniques discussed in section 5.1 extracts base facts and the associated time interval for the base facts. Moreover, we can directly connect a t-fact to a known base fact provided that both appear in the same sentence. For example, the sentence "*After becoming* ***Governor of California****, Schwarzenegger acted in the movie* ***Around the World in 80 Days***." contains three interesting properties, a temporal keyword "*after*", a base

fact and a temporal fact. Our temporal knowledge base has the following facts which implies the phrase "*becoming* ***Governor of California***":

- **f11:** `Arnold_Schwarzenegger` *holdsPoliticalPosition* `Governor_of_California`.
- **f12:** `f11` *startsOnDate* `17-11-2003`.
- **f13:** `f11` *endsOnDate* `03-01-2011`.

Moreover, YAGO knowledge base has the following fact which implies the phrase "*Schwarzenegger acted in the movie* ***Around the World in 80 Days***". However, this YAGO fact does not have any temporal information qualifying it.

- **f31:** `Arnold_Schwarzenegger` *actedIn* `Around_the_World_in_80_Days_(2004_film)`.

The phrase "*becoming* ***Governor of California***" can be mapped to the fact **f11:** `Arnold_Schwarzenegger` *holdsPoliticalPosition* `Governor_of_California` via the techniques described in section 5.2. And the phrase "*Schwarzenegger acted in the movie* ***Around the World in 80 Days***" can be mapped to the fact **f31:** `Arnold_Schwarzenegger` *actedIn* `Around_the_World_in_80_Days_(2004_film)` via tools such as PROSPERA [NTW11] which has pattern statistics for base facts. The idea is to connect t-facts to base facts, thus base facts would be qualified with temporal information.

In this section we concentrate on sentences which have the following patterns:

- <*base phrase*><*temporal keyword*><*temporal phrase*>
- <*temporal keyword*><*temporal phrase*><*base phrase*>

temporal keyword may have the polarity {-1, 0, 1}, where -1=before, 0=during and 1=after. The polarity is used to determine the suitable relation which will be used to connect a t-fact to a known base fact. Possible relations are, `happenedBefore`, `happenedDuring`, `happenedAfter`. The *main phrase* is mined for possible entity pair(s). All base facts containing entities in *main phrase* are checked by using PROSPERA patern statistics to see whether the *main phrase* maps to a known base fact. If PROSPERA returns a base fact for main phrase, then the base fact and t-fact will be connected to each other via the relation determined by the polarity of temporal keyword. Therefore, the base fact will be enriched by a temporal dimension. As an example, the fact **f31:** `Arnold_Schwarzenegger` *actedIn* `Around_the_World_in_80_Days_(2004_film)` will be connected to the fact **f11:** `Arnold_Schwarzenegger` *holds Political Position* `Governor_of_California` as **f39:** `f31` *happenedAfter* `f11` .

Algorithm 3 Connecting T-facts to Known Base Facts

```
1: procedure CONNECT_T_FACTS_TO_BASE_FACTS(temporal sentence ts )
2:     P ← ∅                                              ▷ set of new facts
3:     F ← ∅                                              ▷ set of new existing facts
4:     cf ← CREATE_CANONICAL_FORM(ts)
5:     mp ← GET_MAIN_PHRASE(cf)
6:     tp ← GET_TEMPORAL_PHRASE(cf)
7:     tfact ← GET_TFACT_FROM_DICTIONARY_BASED_SYSTEMS(tp)
8:     polarity ← GET_POLARITY(cf)
9:     R ← DETERMINE_RELATION(polarity)
10:    EP ← GET_ENTITY_PAIRS(mp)
11:    for all entitypair ∈ E do                          ▷ for each entity pair
12:        F ← GET_FACTS_FROM_YAGO(entitypair)
13:        for all f ∈ F do                               ▷ for each fact
14:            f2 ← PROSPERA(f, mp, entitypair)
15:            F ← F∪ CREATE_FACT(f2, R, tfact)
16:    return F
```

5.3 Conclusion

In this section, we split the task of temporal fact extraction from free text into two subsections, temporal fact extraction by normalizing explicit dates and temporal fact extraction by normalizing implicit dates. In former, we used T-YAGO to bootstrap the induction of patterns for a set of given relation . Then, those automatically generated patterns are used to extract base facts and the temporal information to reify base facts. In second subsection, we created two dictionary based systems in order to detect and normalize implicit temporal information in free text. In other words, we developed systems which can detect the existing t-facts appearing in freetext as form of an adverbial clause. At the end of the section, we investigated methods to connect t-facts to known base facts, hence, known base facts will be given a temporal dimension.

Chapter 6

Experiments and Evaluation

This chapter focuses on the experiments conducted for the evaluation of our approaches that are introduced and explained in Chapters 4 and 5 and the overall insights gained from the experiments. There are two kinds of experiments, the experiments about temporal extraction from semi structured data and temporal extraction from free text. In the first one we aim to build a temporal ontolgy called T-YAGO. In the second one, we extend our extraction for free text. These two main experiments are divided into sub-experiments. Each main experiment has a different data collection policy in order to choose the data which is appropriate for this specific experiment. The experiments were conducted on a standard desktop machine with a 3GHz CPU and 3,5GB RAM. We used PostgreSQL Database system to store the facts.

This chapter is organized as follows: Section 6.1 discusses the evaluation measures used. Section 6.2 describes the experiments and the results of extraction from semi structured text. In Section 6.3, the experiments and the results about free text are presented.

6.1 Evaluation Measures

Here, we introduce the measures used to evaluate the effectiveness of our approaches: precision and recall; two metrics widely used in information retrieval.

6.1.1 Precision and Recall

When using precision and recall, the set of possible labels for a given instance is divided into two subsets, "*relevant*" for the purposes at hand or "*irrelevant*". Recall is then computed as the ratio of correct instances among all instances that actually belong to

the relevant subset, while precision is the fraction of correct instances among those that the algorithm decides as relevant. Precision can be seen as a measure of correctness, whereas recall is a measure of completeness. Both can be defined formally as follows.

Definition 1. (Precision) Let D be a set of documents, $R \subseteq D$ be the subset of relevant documents with respect to a query q, $Q \subseteq D$ be the set of documents retrieved for the query. Then, precision is the fraction of retrieved documents that are relevant to the search:

$$Precision = \frac{|R \cap Q|}{|Q|}. \tag{6.1}$$

Definition 2. (Recall) Let D be a set of documents, $R \subseteq D$ be the subset of relevant documents with respect to a query q, $Q \subseteq D$ be the set of documents retrieved for the query. Then, recall is the fraction of the documents that are relevant to the query that are successfully retrieved.

$$Precision = \frac{|R \cap Q|}{|R|}. \tag{6.2}$$

In the evaluation of the correctness of temporal facts, we are interested in whether an extracted fact is correct. Actually, this turns into an evaluation of a classification task. In this context, precision and recall are calculated differently, since the number of possible temporal facts that have to be extracted is unknown. We use the notion of *true positives* as correctly extracted temporal facts and *false positives* as the extracted temporal facts which are incorrect. Then, we define precision as the fraction of correct temporal facts among all extracted facts.

$$Precision = \frac{true\ positives}{true\ positives + false\ positives}. \tag{6.3}$$

And the recall is defined as the number of all extracted temporal facts.

$$Recall = true\ positives + false\ positives. \tag{6.4}$$

In practice, precision and recall are related to each other by a trade-off. High precision usually leads to low recall and vice versa. For this reason, the correctness of a system can be evaluated as the ratio of precision and recall.

Precision is evaluated by manually assessing the correctness of randomly sampled facts.

6.2 Extraction from Semi Structured Text

Here, we present the results of the experiments conducted for harvesting semi-structured text from Wikipedia articles. We used the temporal facts extracted from semi-structured

Data	Size
Articles (redirect pages are included)	6,776,204
Articles with infoboxes	1,338,838
Distinct Infobox Types	3837

TABLE 6.1: Statistics about the Wikipedia dump used in experiments

text to create the knowledge base T-YAGO. We ran our extractors over infoboxes and categories of Wikipedia articles. The experiments and the data collection are explained in the following sections.

6.2.1 Data Collection

We downloaded the English version of the Wikipedia XML dump which is a 23GB text file. Each article in the XML file starts with a "<*page*>" tag. Inside the "<*page*>" tag, all textual information (infoboxes, categories, free text) about the article appears in the "<*text*>" tag. Some properties of the Wikipedia dump that we used are given in Table 6.1.

6.2.2 Harvesting Infoboxes

First of all, as we have already seen, there are many types of infobox templates and the distribution of infobox templates used in the articles is not uniform. In other words, most of the templates are used only in few articles. Therefore, we focus only on some templates which cover the majority of the articles. The distribution of infobox templates used in articles is shown in Figure 6.1.

When harvesting infoboxes, we focused on articles about named events. In order to extract the named events from infoboxes, we developed rules for correct extraction. These rules check the infobox type and decide whether it is a pre-defined event type, such as war, battle, conference, etc. We invoke these extraction rules as a template classifier as well, since they decide the type of the template of the infobox. Some of these rules are shown in Table 6.2.

More than 100,000 events are extracted from infoboxes and their precision is shown in Table 6.3.

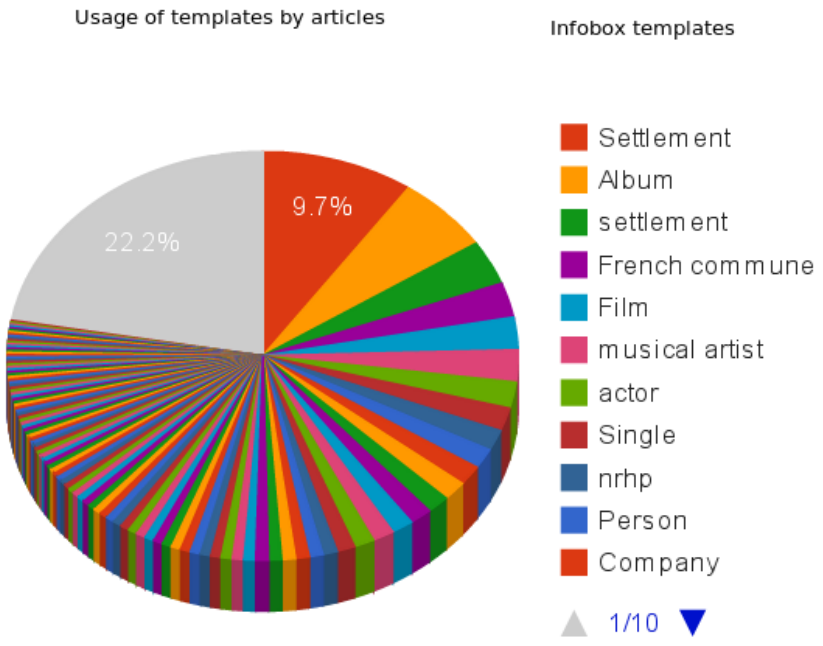

FIGURE 6.1: The distribution of infobox templates used in articles. The gray area (22.2 %) resembles the other templates which forms the long tail of the distribution.

6.2.3 Harvesting Categories

In addition to temporal facts extracted from Wikipedia infoboxes, we also extracted high amount of temporal facts from wikipedia categories. For the sake of re-usability, we built a database of Wikipedia categories which contains 534765 categories. By using regular expressions for dates, we got 60582 categories which contain temporal information. We evaluated randomly choosen 200 temporal categories and the extraction of temporal categories from full set of categories has 98% precision. See Table 6.5. These categories are harvested via manually created rules discussed in Chapter 4. The evaluation of the temporal fact extraction from categories is shown in Table 6.6. Table 6.7 shows the distribution of temporal facts for different types of categories. Moreover, the statistics about temporal categories and about the temporal facts extracted from them are visualized in Figure 6.2.

From Table 6.1, it is seen that most of the articles do not have any infoboxes. Neverthelss, there are more than 1.3 million articles having infoboxes. The latest Wikipedia dump

Infobox Type	**Attribute having the temporal information**
Military Conflict	date
Election	election_date
Treaty	date_signed
Olympic event	dates
Historical Event	Date
Film	released
Football match	date
UN resolution	date
MMA event	date
Wrestling event	date
Music festival	dates
Awards	year
Attack	date
Accident	dates
Swimming event	dates
Competition	year
Congress	start|end
Tournament	fromdate|todate

Table 6.2: Infobox rules

Randomly sampled facts	**Precision**
200	90.0%

Table 6.3: Precision of the event facts

Features	**Size**
Full categories set	534765
Temporal categories set	60582

Table 6.4: Size information about categories

Evaluated Categories	**Precision**
200	98

Table 6.5: Evaluation of the extraction of temporal categories from the full set of categories

Evaluated Facts	Precision
240	93,2%

TABLE 6.6: Evaluation of temporal facts extracted from categories

Temporal Category Types	Number of Categories	Number of Extracted Temporal Facts
Categories about disasters	1001	1477
Categories about musicals	570	2938
Categories about accidents	635	4864
Categories about laws	712	6520
Categories about artistic works	1626	7998
Categories about discoveries	1341	13078
Categories about political events	3265	23172
Categories about constructions	698	22267
Categories about books	1759	39818
Categories about (dis)establishments	10895	174649
Categories about movies	1670	254122
Categories about births and deaths	5904	941281
Total	30076	1492184

TABLE 6.7: Statistics about the temporal categories and about the temporal facts extracted from them

has 3837 distinct infobox types and few of them are used by most of the articles. Thus, the most frequently used infoboxes can be determined and appropriate rules can be declared to harvest these infoboxes. Graph 6.1 shows that 88% of (1 million) articles having infoboxes use around 100 distinct infobox templates. Since, each infobox template has around 20 attribute-pair values, having manually declared rules for 100 templates may yield more than 20 million facts.

In our extraction of named events, we have 90% precision since there are many infoboxes which do not obey Wikipedia template standards. As a result, our extraction rules fail in case of non-standard infoboxes. Sam problem is also valid for extraction from categories. There are many articles which do not belong to the categories assigned for them by Wikipedia authors, which decreases the precision.

Table 6.4 and Table 6.7 show that vast amount of temporal facts can be extracted from Wikipedia categories. Although there are knowledge bases harvesting Wikipedia categories, such as YAGO, DBpedia, they focus on "*isA*" relation to create the taxonomy of the ontology. However, we prove that Wikipedia categories can be harvested for fact extraction apart from "*isA*" relation. More than 1.4 million facts are extracted from

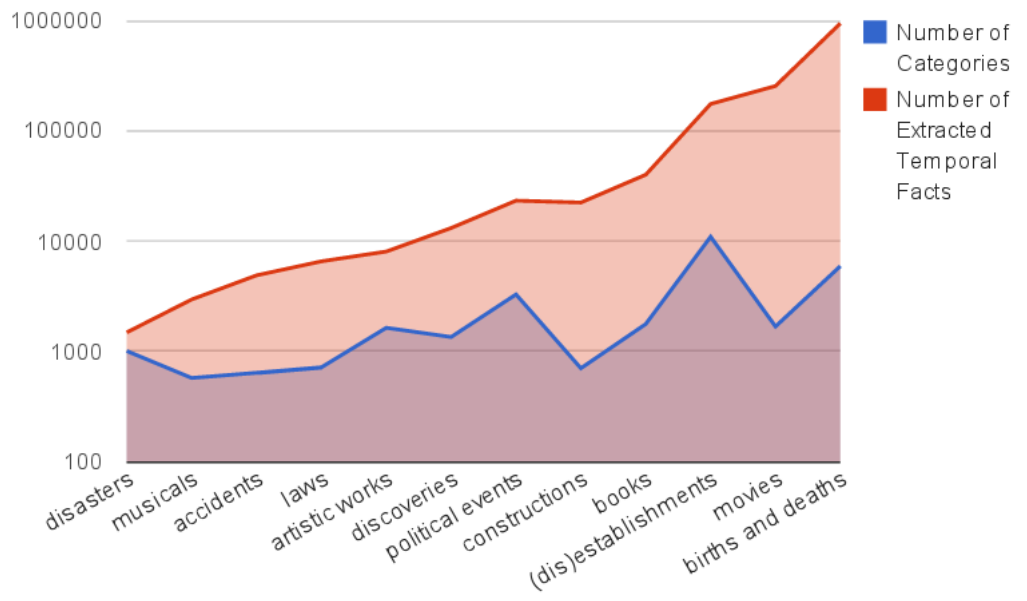

FIGURE 6.2: Statistics about temporal categories and temporal facts shown in log-scale.

around 30 thousand categories, which is encouraging to extract non-temporal facts from categories, as well. For example, the Wikipedia article about *Nicolas Sarkozy* has a category *Candidates for the French presidential election, 2007* which can be harvested to create the fact

- **factID:** *Nicolas_Sarkozy* `participatedIn` *French_presidential_election,_2007.*

6.3 Extraction from Free Text

In this section, we present the experimental results about the temporal fact extraction from free text. We first start by presenting the data collection and the experimental setting, then we continue by showing the experimental results about the pattern-based extraction for exact dates and the dictionary-based extraction for implicit dates.

6.3.1 Data Collection and Experimental Settings

In order to harvest free text, we used a subset of Wikipedia articles as the extraction corpus. We chose 100 Wikipedia articles about politicians. The main reason for which we have chosen the politicians' articles is explained in Section 6.3.2.

Automatically Generated Patterns
<politician> was <political office> from <date> to <date>
<politician> served as <political office> from <date> to <date>
<politician> was the <political office> serving from <date> until <date>
<politician> was inaugurated as <political office> on <date>
<politician> notably served as <political office> from <date> to <date>
<politician> was appointed <political office> on <date>
<politician> was elected <political office> in <date>
<politician> was sworn in as <political office> on <date>
<politician> is a politician who was the <political office> from <date> to <date>

TABLE 6.8: Patterns for the relation `holdsPoliticalPosition`

Relation	**Number of Seeds**	**Input Articles**	**Extracted Facts**	**Precision**
holdsPoliticalPos.	50	100	221	67%

TABLE 6.9: Evaluation of pattern based temporal fact extraction

6.3.2 Pattern-Based Extraction by Normalizing Exact Dates

The process of pattern-based extraction is explained in Chapter 5. In order to start the extraction procedure, we decided to extract facts which have the relation `holdsPoliticalPosition`, since `holdsPoliticalPosition` is one of the relations in YAGO2 which have the least number of facts. For this reason, we focused on the politicians' articles. We collected 100 Wikipedia articles about politicians out of which 50 are selected as seed entities to bootstrap the pattern induction step.

The pattern induction step outputs a set of ranked patterns. Some of those patterns are shown in Table 6.8 in order to give an idea about the structure of the patterns.

The automatically generated patterns are instantiated on the experiment corpus (100 articles) in order to extract base facts and the temporal information qualifying these base facts (reified facts). Table 6.9 shows the precision and recall values of temporal facts extracted by patterns.

6.3.3 Dictionary-Based Extraction by Normalizing Implicit Dates

In this section, we present the results of normalization of implicit dates hidden in temporal adverbial phrases. As it is explained in Chapter 5, the process of dictionary based extraction aims to map temporal phrases to t-facts. This mapping process is completed

Temporal Phrases
during Brezhnev's rule
during the second half of Brezhnev's reign
after the declaration of the Spanish-American War
before the federal government enacted the Sherman Antitrust Act
before being elected to the Senate in 1958
after Harding's death
during the Vietnam War
before World War II
during his childhood
after Bismarck's dismissal

TABLE 6.10: Some interesting temporal phrases found by identification step

System	**Precision**	**Recall**
RDS	31%	45%
CAS	71%-85%	51%

TABLE 6.11: Evaluation of the two dictionary based systems; Rich Dictionary System, Context Aware System.

in three steps; identification, interpretation and normalization, respectively. The identification step aims to detect the temporal phrases in text which are used as input for the interpretation step. We developed two dictionary-based systems for the interpretation step in order to map temporal phrases to t-facts. The normalization step returns the temporal values of t-facts found in the interpretation step by issuing an SQL query to T-YAGO. We present the evaluation of the identification and interpretation step, since the normalization step only returns what exists in T-YAGO.

Table 6.10 shows some interesting temporal phrases found by the identification step. Some of the phrases found by this step do not have temporal information, since the keywords *after, before, during* have also non-temporal meaning, as in the phrases *"after their newspaper, named after him", etc* .

Table 6.11 shows the precision and recall values of the dictionary-based systems (Rich Dictionary System, Context Aware System) utilized for the interpretation stage. Moreover, Figure 6.3 shows the size of the dictionary of the CAS. Recall that RDS loads the entire dictionary of temporal facts which has around 1,4 million temporal facts, whereas CAS dynamically loads temporal facts to the dictionary as it is explained in Chapter 5.

Moreover, we compare our approach with the state of the art temporal information extraction tool TARSQI. We gave some random temporal phrases and dates to TARSQI and also to our date normalizers. Figure 6.4 shows the temporal expressions given to

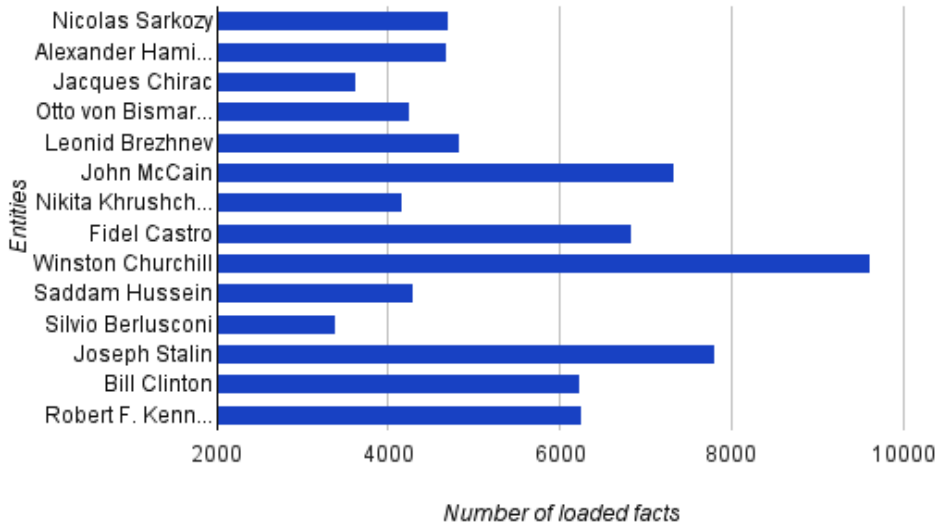

FIGURE 6.3: The number of facts loaded to the dictionary of CAS for different entities

Temporal expressions
He is the 1st French president to have been born after World War II.
He was born on 1 January 1955.
... after the 17 June parliamentary elections...
...first two governments (from May 2002 to March 2004)
It continued from November 23rd to 24th 1998.
...after the death of the incumbent mayor Achille Peretti...
During the 33rd G8 summit in Heiligendamm, Sarkozy set ...

FIGURE 6.4: The temporal expressions given to TARSQI and the correctly normalized expressions are colored blue.

TARSQI and the correctly normalized expressions are colored blue. Although TARSQI is supposed to extract exact dates such as "*1 January 1955*", "*from May 2002 to March 2004, etc.*" , it cannot extract these dates possibly due to bugs. Figure 6.5 shows same temporal expressions given to our regular expression based date normalizer which annotates all exact dates correctly. And Figure 6.5 shows the date normalization produced by our exact date normalizer + implicit date normalizer.

Temporal expressions
He is the 1st French president to have been born after World War II.
He was born on 1 January 1955.
... after the 17 June parliamentary elections...
...first two governments (from May 2002 to March 2004)
It continued from November 23rd to 24th 1998.
...after the death of the incumbent mayor Achille Peretti...
During the 33rd G8 summit in Heiligendamm, Sarkozy set ...

FIGURE 6.5: The temporal expressions given to our exact date normalizer and the correctly normalized expressions are colored blue.

Temporal expressions
He is the 1st French president to have been born after World War II.
He was born on 1 January 1955.
... after the 17 June parliamentary elections...
...first two governments (from May 2002 to March 2004)
It continued from November 23rd to 24th 1998.
...after the death of the incumbent mayor Achille Peretti...
During the 33rd G8 summit in Heiligendamm, Sarkozy set ...

FIGURE 6.6: The temporal expressions given to our exact date normalizer + implicit date normalizer and the correctly normalized expressions are colored blue.

6.3.4 Summary

In this section, we evaluated effectiveness our approaches by providing experiments. The extraction from semi-structured text is promising in terms of precision and recall. However, the recall still remains as an open problem in extraction from free text. The possible reasons of low recall are difficulty of coreference resolution, named entity recognition and entity disambiguation. Although the heuristics that we use to solve these problems can work fairly well, a complete and effective framework is needed to overcome these challenges.

Chapter 7

Conclusions and Future Directions

7.1 Conclusion

In this thesis, we have presented a framework to extract temporal facts from semi-structured part of Wikipedia articles. We also introduced an automatic pattern induction method to create patterns for given relations and to rank them based on frequency statistics of patterns. We developed algorithms to extract temporal facts from free-text by normalizing explicit dates. Moreover, we introduced the idea of normalizing implicit dates occurring in adverbial phrases by utilizing our temporal knowledge base. Additionally, we proposed a novel method to introduce temporal dimension to already existing ontological facts.

7.2 Future Directions

Our work has been developed by using baseline solutions and has space for further improvements. We see some possible directions of future works as follows:

- **Developing a full-fledged temporal ontology.** In our current work, we build T-YAGO on YAGO architecture via extending the knowledge representation model and adding new temporal predicates. However, building a new ontology which is aware of temporal dimension of entities and facts, the interactions of facts, the ordering and the causality of events, time-line of events, life stories of important entities (such as famous persons, incorporations, organizations, etc.) still remains

untouched. Such an ontology requires well defined semantics for representation of concepts, as well as requiring a special query language to retrieve information.

- **Exploring multimedia information with temporal information.** Another interesting extension to a temporal ontology is integrating multimedia information, such as images. For example, using different images to visualize the timeline of the Second World War will be quite appealing.

- **Newswire and other Web sources as extraction corpus.** We currently use Wikipedia articles for extraction. However, we further aim to extend our extraction to the news archives and other qualitative web sources, since they contain high amount of temporal information.

- **Improving recall.** Although our experiments show satisfying precision, the recall still is the bottle neck during harvesting the free text. One of the main reason of low recall is difficulty of coreference resolution, named entity recognition and entity disambiguation problems. We plan to incorporate advanced techniques of coreference resolution, named entity recognition and entity disambiguation to our extractors.

List of Figures

2.1 New York Times article annotated by using TARSQI 11

3.1 An excerpt of YAGO RDF graph. 17
3.2 YAGO data types. 17
3.3 T-YAGO date data type. 19
3.4 T-YAGO representation of time points and durations. "*On Time Point, Since Time Point, Until Time Point*" are associated with the relations `happenedOnDate, startedOnDate, endedOnDate`, respectively. 20

4.1 Categories belonging Nicolas Sarkozy's Wikipedia article 28
4.2 The process of harvesting temporal facts and events from Wikipedia categories 31

5.1 The linkage found for the sentence "*After his re-election as President of the French Republic, Chirac appointed Sarkozy as French Minister of the Interior.*" by The Link Grammar Parser 42
5.2 The constituent tree produced from the linkage shown in figure 5.1 43
5.3 An excerpt from Wikipedia Hyperlink Graph 45
5.4 The graph showing the entity Nicolas_Sarkozy and its first neighbours . . 46

6.1 The distribution of infobox templates used in articles. The gray area (22.2 %) resembles the other templates which forms the long tail of the distribution. 52
6.2 Statistics about temporal categories and temporal facts shown in log-scale. 55
6.3 The number of facts loaded to the dictionary of CAS for different entities 58
6.4 The temporal expressions given to TARSQI and the correctly normalized expressions are colored blue. 58
6.5 The temporal expressions given to our exact date normalizer and the correctly normalized expressions are colored blue. 59
6.6 The temporal expressions given to our exact date normalizer + implicit date normalizer and the correctly normalized expressions are colored blue. 59

List of Tables

4.1 Wikipedia Markup Language . . . 24
4.2 Event infobox types and attributes. An attribute map contains all the attributes of event infoboxes. . . . 26
4.3 Category-relation map . . . 30

5.1 Patterns for the relation `holdsPoliticalPosition` . . . 39
5.2 Patterns for candidate event detection . . . 41
5.3 Convertions from facts to noun phrases based on the relation . . . 44

6.1 Statistics about the Wikipedia dump used in experiments . . . 51
6.2 Infobox rules . . . 53
6.3 Precision of the event facts . . . 53
6.4 Size information about categories . . . 53
6.5 Evaluation of the extraction of temporal categories from the full set of categories . . . 53
6.6 Evaluation of temporal facts extracted from categories . . . 54
6.7 Statistics about the temporal categories and about the temporal facts extracted from them . . . 54
6.8 Patterns for the relation `holdsPoliticalPosition` . . . 56
6.9 Evaluation of pattern based temporal fact extraction . . . 56
6.10 Some interesting temporal phrases found by identification step . . . 57
6.11 Evaluation of the two dictionary based systems; Rich Dictionary System, Context Aware System. . . . 57

Appendix A

Classification of Temporal Expressions

Conceptual Category	Temporal Expression	TimeML Support	TimeML Type	Found by TARSQI
Date	On 01/25/2009, the second of December, "Friday, October 1, 1999", October of 1963	Yes	Date	No
Date	Tuesday 18th	Yes	Date	No
Date	On 1 January 1985	Yes	Date	No
Date	May 2001, October 1998,	Yes	Date	No
Part of Day	In the morning, in the evening	Yes	Time	No
Part of Day	This morning, this evening, tonight, this afternoon, noon, midnight,	Yes	Time	Yes
Weekdays	Wednesday, Friday. Thursday, Saturday night,	Yes	Date	Yes
Time of Day	At 5 am, at 4 pm	Yes	Time	Yes
Conjuction	From 'October 20 to October 30', January- May, November 23rd to 24th 1998	-	-	No
Conjuction	1998-1999, Wednesday-Friday	-	-	No
Conjuction	From October to November, from 1998 to 1999	Yes	Date	Yes
Relative Date and Time	Yesterday, today, tomorrow, next week, next year, in the past month, last century, next century, last decade, last winter, this summer, next fall, this weekend, this past autumn	Yes	Date	Yes
Relative Date and Time	two weeks from next Tuesday	Yes	Date	No
Relative Date and Time	three days, the last four days, the past 3 years, the third consecutive month, two entire days	Yes	Duration	Yes
Relative Date and Time	Three days ago, two decades ago, 2 days before yesterday, recent months	Yes	Duration	No
Relative Date and Time	10 days earlier	-	-	No
Special Days	Valentine day, Xmas, Christmas, mother day, Halloween	Yes	Date	Yes

Conceptual Category	**Temporal Expression**	**TimeML Support**	**TimeML Type**	**Found by TARSQI**
Fuzzy Time	At that time, at the same time, the best second quarter ever	-	-	No
Fuzzy Time	Third quarter, nearly four decades	Yes	Duration	No
Event-dependent Time	During the World War 2nd , after German Reunification, before the 33rd G8 summit	-	-	No
Recurring Times	Each Wednesday, every year, each week	Yes	Set	Yes
Recurring Times	every October	Yes	Set	Yes

Bibliography

[ABK+07] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. *The Semantic Web*, pages 722–735, 2007.

[ABL+07] S. Auer, C. Bizer, J. Lehmann, G. Kobilarov, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *Proc. ISWC/ASWC*, LNCS 4825. Springer, 2007.

[AG00] Eugene Agichtein and Luis Gravano. Snowball: extracting relations from large plain-text collections. In *Proc. 5th ACM Conference on Digital Libraries (DL '00)*, pages 85–94, New York, NY, USA, 2000. ACM.

[AGBY07] O. Alonso, M. Gertz, and R. Baeza-Yates. On the value of temporal information in information retrieval. In *ACM SIGIR Forum*, volume 41, pages 35–41. ACM, 2007.

[Agi05] Eugene Agichtein. Scaling information extraction to large document collections. *IEEE Data Eng. Bull*, 28:3–10, 2005.

[All83] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[BC10] A. Balmin and E. Curtmola. WikiAnalytics: Ad-hoc querying of highly heterogeneous structured data. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 1145–1148. IEEE, 2010.

[BCS06] G.D. Bader, M.P. Cary, and C. Sander. BioPAX–biological pathway data exchange format. 2006.

[BCSW07] Holger Bast, Alexandru Chitea, Fabian Suchanek, and Ingmar Weber. Ester: Efficient search in text, entities, and relations. In *Proc. SIGIR*, Amsterdam, Netherlands, 2007. ACM.

[Ber10] K.L. Berberich. *Temporal search in web archives.* PhD thesis, Universitätsbibliothek, 2010.

[BF06] Thorsten Brants and Alex Franz. Web 1T 5-gram Version 1. *Linguistic Data Consortium, Philadelphia*, 2006.

[BG04] D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema: W3C Recommendation 10 February 2004. 2004.

[BM+01] P.V. Biron, A. Malhotra, et al. XML schema part 2: Datatypes. *W3C recommendation*, 2:2–20010502, 2001.

[BPX+07] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. In *Proc. EMNLP-CoNLL 2007*, pages 858–867, 2007.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[Bri99] S. Brin. Extracting patterns and relations from the world wide web. *The World Wide Web and Databases*, pages 172–183, 1999.

[CDSE05] Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. Knowitnow: Fast, scalable information extraction from the web. In *HLT/EMNLP* [DBL05].

[CL01] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2001.

[Cru86] D. Alan Cruse. *Lexical Semantics*. Cambridge University Press,, Cambridge, UK, 1986.

[DBL05] *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*. The Association for Computational Linguistics, 2005.

[DSE07] Doug Downey, Stefan Schoenmackers, and Oren Etzioni. Sparse information extraction: Unsupervised language models to the rescue. In *In Proc. of ACL*, pages 696–703, 2007.

[ECD+04a] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. Web-scale information extraction in KnowItAll: Preliminary results. In *Proc. WWW 2004*, pages 100–110, 2004.

[ECD+04b] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of the 13th international conference on World Wide Web*, pages 100–110. ACM, 2004.

[FS96] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. ICML 1996*, pages 148–156, San Francisco, 1996. Morgan Kaufmann.

[GBM06] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Automatic discovery of part-whole relations. *Comput. Linguist.*, 32(1):83–135, 2006.

[GC03] David Graff and Christopher Cieri. English Gigaword. *Linguistic Data Consortium, Philadelphia*, 2003.

[GCU05] Zhiguo Gong, Chan Wa Cheang, and Leong Hou U. Web query expansion by WordNet. In *Proc. DEXA 2005*, volume 3588 of *LNCS*, pages 166–175. Springer, 2005.

[HL93] B.L. Humphreys and DA Lindberg. The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association*, 81(2):170, 1993.

[HNP09] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24:8–12, 2009.

[HSB+] J. Hoffart, F.M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, and G. Weikum. YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages.

[Kin05] D. Kinzler. Wikisense-mining the wiki. In *Proceedings of Wikimania*, volume 5, 2005.

[KM10] O. Kolomiyets and M.F. Moens. KUL: Recognition and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 325–328. Association for Computational Linguistics, 2010.

[Len95] D.B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[LK04] Mirella Lapata and Frank Keller. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of nlp tasks. In *HLT-NAACL*, pages 121–128, 2004.

[LNYW10] X. Liu, Z. Nie, N. Yu, and J.R. Wen. BioSnowball: automated population of Wikis. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 969–978. ACM, 2010.

[LW10] X. Ling and D.S. Weld. Temporal information extraction. In *Proceedings of the Twenty Fifth National Conference on Artificial Intelligence*, 2010.

[Mil95] G.A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[MWN07] David N. Milne, Ian H. Witten, and David M. Nichols. A knowledge-based search engine powered by Wikipedia. In *Proc. CIKM 2007*, New York, NY, USA, 2007. ACM.

[NP01] I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9. ACM, 2001.

[NTW10] N. Nakashole, M. Theobald, and G. Weikum. Find your advisor: robust knowledge gathering from the web. In *Procceedings of the 13th International Workshop on the Web and Databases*, pages 1–6. ACM, 2010.

[NTW11] N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 227–236. ACM, 2011.

[Ont08] O.B. Ontologies. The Gene Ontology http://www. geneontology. org. *Accessed on February*, 21, 2008.

[PK04] M. Poesio and M.A. Kabadjov. A general-purpose, off-the-shelf anaphora resolution module: Implementation and preliminary evaluation. In *Proc. LREC*. Citeseer, 2004.

[PRH04] Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. Towards terascale knowledge acquisition. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 771, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[PV09] J. Pustejovsky and M. Verhagen. SemEval-2010 task 13: evaluating events, time expressions, and temporal relations (TempEval-2). In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 112–116. Association for Computational Linguistics, 2009.

[QKC04] L. Qiu, M.Y. Kan, and T.S. Chua. A public reference implementation of the rap anaphora resolution algorithm. *Arxiv preprint cs/0406031*, 2004.

[Ril93] Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In *AAAI*, pages 811–816, 1993.

[Sar08] S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

[SIW06] F.M. Suchanek, G. Ifrim, and G. Weikum. LEILA: Learning to extract information by linguistic analysis. In *Proceedings of the ACL-06 Workshop on Ontology Learning and Population*, pages 18–25, 2006.

[SKW07a] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proc. WWW 2007*, New York, NY, USA, 2007. ACM Press.

[SKW07b] F.M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[SSW09] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. Sofie: a self-organizing framework for information extraction. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 631–640, New York, NY, USA, 2009. ACM.

[ST95] D.D.K. Sleator and D. Temperley. Parsing English with a link grammar. *Arxiv preprint cmp-lg/9508004*, 1995.

[Suc] F.M. Suchanek. Automated Construction and Growth of a Large Ontology.

[Sun92] B.M. Sundheim. Overview of the fourth message understanding evaluation and conference. In *Proceedings of the 4th conference on Message understanding*, pages 3–21. Association for Computational Linguistics, 1992.

[VGS+07] M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, G. Katz, and J. Pustejovsky. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 75–80. Association for Computational Linguistics, 2007.

[VMS+05] M. Verhagen, I. Mani, R. Sauri, R. Knippen, S.B. Jang, J. Littman, A. Rumshisky, J. Phillips, and J. Pustejovsky. Automating temporal annotation with TARSQI. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 81–84. Association for Computational Linguistics, 2005.

[VP08] M. Verhagen and J. Pustejovsky. Temporal processing with the TARSQI toolkit. In *22nd International Conference on on Computational Linguistics: Demonstration Papers*, pages 189–192. Association for Computational Linguistics, 2008.

[VPP+08] Y. Versley, S.P. Ponzetto, M. Poesio, V. Eidelman, A. Jern, J. Smith, X. Yang, and A. Moschitti. BART: A modular toolkit for coreference resolution. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Demo Session*, pages 9–12. Association for Computational Linguistics, 2008.

[WTV+10] Kuansan Wang, Chris Thrasher, Evelyne Viegas, Xiaolong Li, and Bojune (Paul) Hsu. An overview of microsoft web n-gram corpus and applications. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, pages 45–48, Los Angeles, California, June 2010. Association for Computational Linguistics.

[WW07] F. Wu and D.S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50. ACM, 2007.

[WW08] F. Wu and D.S. Weld. Automatically refining the Wikipedia infobox ontology. In *Proceeding of the 17th international conference on World Wide Web*, pages 635–644. ACM, 2008.

[WWA+08] D.S. Weld, F. Wu, E. Adar, S. Amershi, J. Fogarty, R. Hoffmann, K. Patel, and M. Skinner. Intelligence in wikipedia. In *Twenty-Third Conference on Artificial Intelligence (AAAI'08)*, 2008.

[WZQ+10] Y. Wang, M. Zhu, L. Qu, M. Spaniol, and G. Weikum. Timely YAGO: Harvesting, querying, and visualizing temporal knowledge from Wikipedia. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 697–700. ACM, 2010.

[YCB+07] A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead, and S. Soderland. TextRunner: Open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations on XX*, pages 25–26. Association for Computational Linguistics, 2007.

[ZG07] T. Zesch and I. Gurevych. Analysis of the Wikipedia category graph for NLP applications. *TextGraphs-2: Graph-Based Algorithms for Natural Language Processing*, page 1, 2007.

[ZNL+09] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.R. Wen. StatSnowball: a statistical approach to extracting entity relationships. In *Proceedings of the 18th international conference on World wide web*, pages 101–110. ACM, 2009.

[ZSYW08] Q. Zhang, F.M. Suchanek, L. Yue, and G. Weikum. TOB: Timely ontologies for business relations. In *Proceedings of the International Workshop on the Web and Databases (WebDB)*. Citeseer, 2008.