**Universität des Saarlandes**
**Max-Planck-Institut für Informatik**
**AG5**


# Context-Aware Timeline for Entity Exploration

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

Anh Tuan Tran

angefertigt unter der Leitung von / supervised by

Prof. Dr. Gerhard Weikum

betreut von / advised by

Dr. Nicoleta Preda

Shady Elbassuoni

begutachtet von / reviewers

Dr. Martin Theobald

Prof. Dr. Gerhard Weikum

März / March 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

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

Saarbrücken, den 14. März 2011,

(Anh Tuan Tran)

### 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, Saarland University.

Saarbrücken, den 14. März 2011,

(Anh Tuan Tran)

*To my father Duc, my mother Loan, my two sisters Hai Anh and Lan Anh, and my better half Hong Nhung*

- Tuan

# *Abstract*

With millions of articles in multiple languages, Wikipedia has become the de-facto source of reference on the Internet today. Each article on Wikipedia contains encyclopedic information about various topics (people, events, inventions, etc.) and implicitly represents an entity. Extracting the most important facts about such entity will help users to find desired information more quickly and effectively. However, this task is challenging due to the incomplete and noisy nature of Wikipedia articles. This calls for a mechanism to detect and summarize the most important information about an entity on Wikipedia.

This thesis proposes and implements CATE (**C**ontext-**A**ware **T**imeline for **E**ntity Exploration), a framework that utilizes Wikipedia to summarize and visualize the important aspects of entities in a timeline fashion. Such a system will help users to draw quickly an informative picture of an entity (e.g. life of a person, or evolution of a research topic, etc.). The novelty of CATE lies in seeing the entity in different contexts, synchronous with contemporaneous events. In addition, CATE puts the entity in a relationship with other entities, and thus offers a broader portrait about it. In order to efficiently query and visualize the events related to the entity, a number of techniques have been developed, combining information extraction and information retrieval with a novel ranking model. The thesis also discusses several experiments and evaluation results to show the effectiveness of the methods proposed.

# *Acknowledgements*

First and foremost, I would like to thank my advisors Dr. Nicoleta Preda and Shady Elbassuoni, for their invaluable guidance. I feel deeply indebted to their strong advises, encouragements and instructions. They have not only assisted me in completing this thesis, they have also helped me to broaden my attitude towards research, and to develop my personality.

A special note of thanks to Prof. Gerhard Weikum for giving me the opportunity to pursue this thesis under his supervision and for his valuable inputs. I am very grateful to work with his group in the Max-Planck Institute for Informatics. It is a priceless experience to me.

I am grateful to Dr. Martin Theobald to review my master thesis.

I am also very grateful to my friends and classmates for their company and moral support. I learnt a lot from them in last 2 years which is a very important factor for successful completion of my thesis. Special thanks are sent to the colleague students in the lab room 4.11 for the memorable time working with them.

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

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Wikipedia is probably one of the most widely used reference work on the Internet. Not only it archives an enormous collection of articles (17 million articles in over 250 languages, out of which 3.5 million are in English [1]), but it is also growing rapidly which is credited to the active community of thousands of contributors worldwide. Wikipedia provides fairly reliable information due to its unique editing policy. Leveraging two extents of trustworthiness and the order of magnitude, Wikipedia provides an exceptional source for adequate quality information on the Web. Nowadays, Wikipedia has become one of the most authoritative information source on the Web.

Each Wikipedia article describes a single concept or an entity, e.g. people, countries, historical events, etc. Finding information about these entities is a basic need of users of Wikipedia. However, this is not always accomplished because of the diverse and incomplete nature of Wikipedia. More specifically:

1. Articles are written and edited by multiple authors at different moments. Information is often added and organized without a consistent or controlled style. Let us illustrate with the article about Carl Friedrich Gauss [1], a famous German mathematician. Information about Gauss' scientific work in mathematics, astronomy, etc. and information about Gauss' personal life (his education, his family,...) disperse over several sections in the article. This is difficult for users who want to find the most important contributions of Gauss in mathematics, or who want to know about Gauss' personal life.

[1] http://en.wikipedia.org/wiki/Wikipedia
[1] http://en.wikipedia.org/wiki/Carl_Friedrich_Gauss

2. The articles are often incomplete. For example, one of the most popular anecdote about Gauss, the story that he pioneered the field of summation with summing n first integer numbers at the age of 7, is not found in his Wikipedia article.

3. Sometimes when user reads an article, she might be interested in other articles within certain aspect. For example, the user might want to find information about other mathematicians who are closest to Gauss in number theory. Unfortunately, there is no means to find related articles in Wikipedia.

These deficiencies call for an effective mechanism to summarize entities in Wikipedia, taking into account the diversity, noisy nature and incompleteness of Wikipedia articles. The summary should be visualized in a meaningful and appealing interface, enabling users to quickly browse information about the entity.

## 1.2 Our Approach

In this thesis, we aim to build a system that summarizes and visualizes the important aspects of an entity in Wikipedia in a *timeline* fashion. The system, which is called *CATE* (**C**ontext-**A**ware **T**imeline for **E**ntity Exploration), constructs an extended timeline to display different events of an entity (a person, a research topic, an event, an invention, etc.) during its lifespan, taking into consideration its various contexts. For example, the timeline of Gauss shows many notable events during Gauss' life: he was born in Braunschweig in 1777, he studied in the university of Götingen, he was awarded Copley Medal in 1838,... In a nutshell, CATE has the following features:

1. **Event retrieval**. Given an entity, CATE retrieves events of the entity from its Wikipedia article, as well as from other articles over Wikipedia. This results in a more informative picture about the entity.

2. **Contexts**. In CATE, each entity is associated with different contexts. Context is a piece of information which exposes one facet of an entity, and provides a background to explore on one or more aspects of it. We define a context as an object with three attributes: *time*, *location* and *topic*. For example, attributes of Gauss' contexts are:

   - time: 18-th century, 19-th century,...
   - location: Braunschweig, Germany,...
   - topic: number theory, differential geometry, astronomy, mathematics,...

The context enriches the timeline and brings in a comprehensive portrait about the entity.

3. **Relevant entities**. For each entity and context, the timeline exhibits other entities related to the entity within the given context. These related entities help user to gain insights into the position of the entity in the context, as well as its relationships with other entities in Wikipedia. For instance, the timeline of Gauss shows additional information about Legendre or Riemann, two mathematicians whose work is closely related to that of Gauss in the contexts of number theory or differential geometry, respectively.

## 1.3 Problem statement

CATE takes as an input an entity whose article is available on Wikipedia, and confronts the following problems:

1. *Finding the most important events in Wikipedia related to the entity and organizing these events on the timeline* . These events are not always explicitly stated in a single Wikipedia article, but can be scattered over several sources in Wikipedia.
2. *Finding contexts that best describe the many aspects of the entity.* Context provides a common ground for a better understanding of entities and their relationships.
3. *Given a context, finding the most relevant entities to the input entity and with respect to the context.* Finding entities with the awareness of context helps in understanding better the role of the entity and its interaction with other entities in a particular perspective. Since the number of relevant entities can be extremely large, this problem entails: *Finding the top-k relevant entities to a given entity with respect to a given context.*

## 1.4 Contributions

We make the following contributions:

1. We propose the notion of *context* to capture multiple aspects of an entity. We believe that the contexts should play a role in the timeline for the following reasons: 1) they represent an additional source of information about the entity; 2) they provide a background for the entity and explain how it interacts with other entities

within the background; 3) they serve as a means to enable users to focus on events related to one or more contexts of interest.

2. We propose a novel model of entity ranking that is context aware. Our model is based on statistical language models [1].

3. We develop a prototype that integrates knowledge bases, information extraction and information retrieval techniques. Using the RDF standard, our system can be integrated with other other knowledge-based systems [2, 3] with minimal effort.

4. We develop an information extraction framework that extracts several structured data from Wikipedia to enrich the knowledge base.

5. We develop an end-to-end system to visualize results in an intuitive and interactive way. Our idea of multi-faceted browsing of an entity timeline can be employed in other applications as well [4].

## 1.5 Outline

The rest of the thesis is organized as follows. In Chapter 2, we discuss the related work. In Chapter 3, we explain the data model used in CATE. Chapter 4 describes the system architecture of CATE in general. In Chapter 5, we demonstrate our information extraction framework. In Chapter 6, we explain the ranking methods used to retrieve relevant entities and contexts in CATE. Chapter 7 discusses the implementation of CATE from the technical point of view. In Chapter 8, we presents evaluational experiments and results. Finally, Chapter 9 makes a conclusion and proposes some extending directions of our work.

# Chapter 2

# Related Work

Our work involves a wide variety of lines of research in the areas of knowledge discovery and information retrieval. In this chapter, we discuss recent approaches in information extraction from Wikipedia (Section 2.1 and 2.2), in knowledge visualization (Section 2.3), as well as in related subjects of rankings in information retrieval (Section 2.4).

## 2.1 Wikipedia-based Information Extraction

Information extraction (IE) is the process of generating structured data from unstructured or semi-structured information sources (e.g., finding relationships between organizations and locations from a natural language document). Due to its high quality and unique structure, Wikipedia has recently attracted much attention in information extraction. Wikipedia-based information extraction tasks make use of many different building blocks of a Wikipedia article, for instance:

- **Infoboxes**: A special table-style section in each Wikipedia article summarizing basic facts mentioned in the article.
- **Categories**: A special page created by users to group similar articles in some aspects. Each page can be associated with more than one category. Categories can also be grouped further into super-categories, thereby creating a hierarchical network of categories
- **Disambiguation pages**: For an ambiguous title, Wikipedia has a special page guiding users with possible meanings of the title, nad links to the corresponding articles. Disambiguation pages are valuable for entity disambiguation task in IE.

- **Hyperlinks**: Wikipedia enables users to put a hyperlink from a phrase in an article text to a page defining the meaning of the phrase. This forms a network of links between articles with phrases anchored for each link. This network is a valuable source of various tasks in IE, e.g. named entity recognition or semantic relatedness.
- ...

In this section, we only discuss approaches in Wikipedia IE that are related to our work. The first one is information extraction from infoboxes. Due to rich semi-structured data in infoboxes, infoboxes has attracted lots of effort in IE recently with impressive results. Successful endeavors include the projects YAGO at the Max-Planck Institute for Informatics [2], DBPedia at the FU Berlin [5], and KOG at the university of Washington [6]. Most of them result in a large knowledge base, representing information extracted in first-order logic style language. For example, the YAGO team has built a huge ontology with more than 1.7 million entities and 15 million facts [1]. Beside infoboxes, YAGO also mines leaf categories in the categories network, and integrates with another reliable sources such as Wordnet [7], a large lexicon based on experts, for quality boosting. In our work, we utilize YAGO to improve the precision of the events extracted (Chapter 3).

Another approach involves tapping Wikipedia categories for knowledge acquisition. The most related work to ours is the work by Nastase et al. [8]. The authors exploit the patterns in category titles to decode the concepts in the categories, then use the category network to induce the relations between concepts. One interesting point is that the authors attempt to derive relations from the common pattern [$X$ by $Y$] in category names. Each sub-category of a category following this pattern corresponds to one instance of the class $Y$, which helps in building the $isA$ taxonomy for the concept $Y$. As illustrated in Figure 2.1, the $isA$ relationship between the concept Miles Davis and the concept Artist are derived from the fact that Miles Davis Albums is the sub-category of the category Albums By Artist. This heuristic works well and yields the large taxonomy with acceptable precision [1]. However, this approach has the following drawbacks:

- category name patterns are hand-crafted and not scalable,
- it handles Wikipedia categories in syntactic level, and merely extracts the words rather than a named entity,

[1] In the newer version (YAGO2), the authors claimed to extract more than 10 million entities and 80 million facts

[1] The author claimed to have 9 million facts in the taxonomy, with precision ranging from 84% to 98% depending on relation names. URL: http://www.h-its.org/english/research/nlp/download/wikitaxonomy.php

- it does not address the issue of significance of patterns in a category. For example, categories that have a handful of sub-categories will not be reliable for extracting patterns as other well-populated categories.

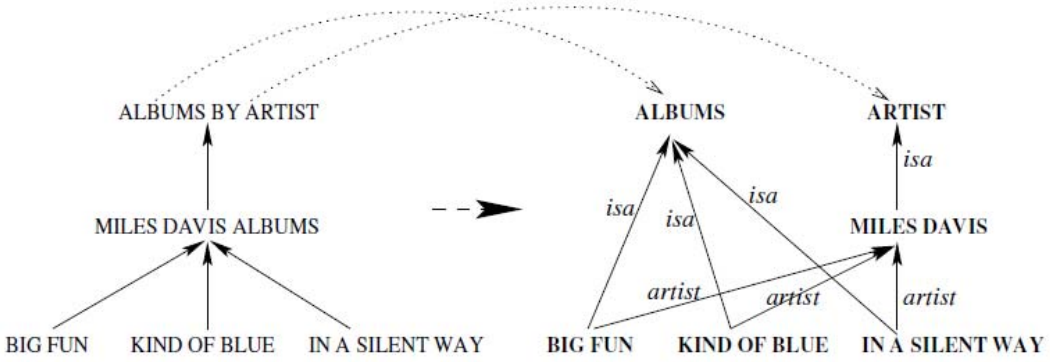


**Figure 2.1:** Relations inferred from [$X$ by $Y$]-patterned categories, from [8]

Following the same line of research on Wikipedia categories, but in another avenue, Schönhofen et al. [9] aim to annotate Wikipedia articles with labels or categories on Wikipedia that most characterize the content of the articles. The idea is that, for a given article and a category, the authors define the matching score between the article and each individual article under the category based on the number of words they share. The matching score between the category and the article is then an aggregation of all matching scores between member articles and the article of interest. However, this approach does not use the links between Wikipedia articles, and it handles words merely in syntactic level.

## 2.2 Event and Temporal Fact Extraction

Event and temporal fact extraction can be seen as a special type of information extraction where results are equipped with temporal information. Since time information is a fist-class citizen in event extraction, much of existing work [10, 11] tries to boost result through detecting patterns. In TimeML project [11], the authors attempt to specify possible types of temporal expressions in natural language, thereby supporting grammar-based extraction approaches. TARSQI Toolkit [10] goes further by introducing a model to represent *underspecified time intervals* , i.e. time intervals which are unknown but can be bounded by inferring from other time information (for example, one event happens after the other event). TARSQI also enables inference of temporal information using

ontology reasoning or learning from meta-data (such as publication date of the document). TARSQI is developed to extract events in the form of temporal relations from semi-structured data using pattern-based algorithms, and from free text using deep parsing techniques based on link grammar.

With the proliferation of Wikipedia as a high-quality source for IE, there is a trend of bringing time dimension to existing Wikipedia-based IE work. Timely YAGO [12] (T-YAGO) is probably one of the earliest work in this line. The authors extend the YAGO knowledge base by introducing new predicates to capture the temporal aspects of existing facts. Events and temporal facts are extracted from Wikipedia infoboxes, categories and lists embedded in articles, and are integrated into the YAGO base. As the first attempt, the authors have extracted 300,000 facts in the sport domain, of which 70,000 facts are from Wikipedia categories and lists present in articles.

## 2.3 Knowledge Visualization

Visualizing an entity (extracted events, facts of the entity, relationships with other entities) enables users to understand the entity better, as well as to gain insight into the knowledge base. Knowledge visualization is an active research field. Here we only consider work that visualizes knowledge in a timeline fashion.

### 2.3.1 Yahoo! Correlator Timeline

The closest work to CATE is the Correlator system [13] from Yahoo! Research. It allows exploring an entity from many perspectives (Figure 2.2 and 2.3). The Correlator also operates on top of Wikipedia. It is mainly used to correlate a set of entities, and derives relationships between them based on the Wikipedia articles and categories that contain them (Figure 2.2). The Web search engine results are also used to compute the entity relevance. In addition, the Correlator provides a timeline to display the various events of entity in the chronological order. Figure 2.3 shows example events in case of Carl Friedrich Gauss.

However, the Correlator has the following disadvantages as compared to our approach:

- it only uses the Wikipedia article of the entity as the data source for timeline visualization,
- it extracts events using shallow parsing technique, resulting in a very low-quality results,

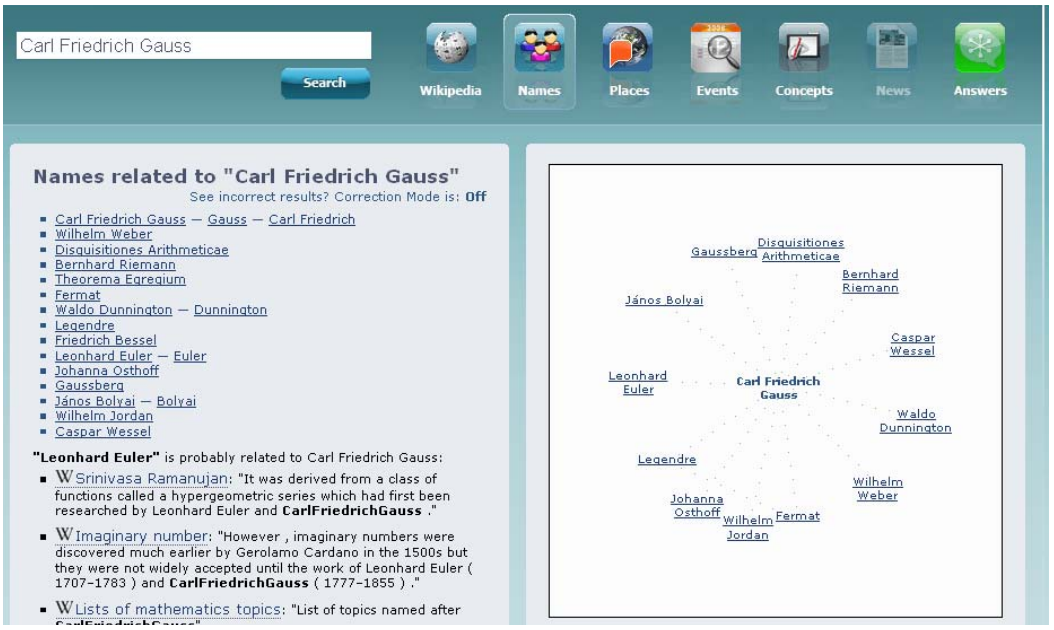


**Figure 2.2:** Snapshot of Yahoo! Correlator for the query "Carl Friedrich Gauss" in named entities view

- it does not leverage the context information

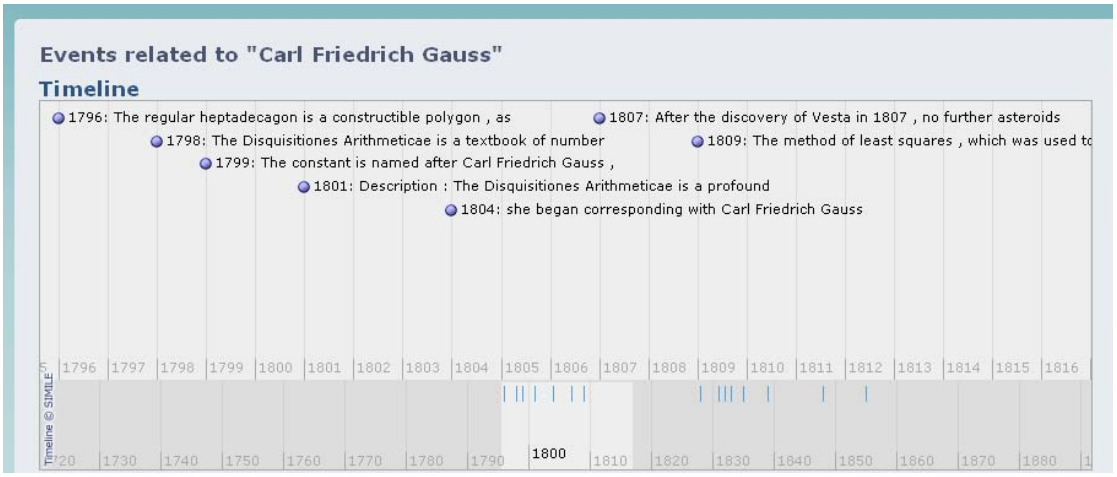


**Figure 2.3:** Snapshot of Yahoo! Correlator for the query "Carl Friedrich Gauss" in events view

In CATE, we extract events of an entity not only from its very Wikipedia article, but also from other articles over the entire Wikipedia. In addition, we utilize the knowledge base to improve the quality of extracted events. Finally, as stated in Subsection 1.2, we introduce the context in CATE in order to provide a more comprehensive portrait of an entity.

### 2.3.2 E-Culture

Another related work to ours is the project MultimediaN E-Culture [14]. It is a software tool [2] providing multimedia faceted-search on distributed data collections of cultural heritage objects. E-Culture clusters entities in the data collection into categories or classes (e.g., art works, artists, etc.). Given a certain category, the system retrieves a set of entities that are most relevant to the category, and shows the evolution of the category over the timeline. E-Culture has somewhat a similar approach to CATE in visualizing entities in a timeline fashion and providing context-aware browsing (here context is the categories of entities in collections). However, it explores the knowledge from the context point of view, whereas in CATE we explore the knowledge from the entity point of view.

## 2.4 Entity Ranking

In contrast to classical information retrieval, where documents are returned in response to a keyword-based query, entity ranking aims to return lists of entities (typically in structured formats) to satisfy certain information needs. This offers a more convenient way to fulfill user interest and has gained increasing attention recently. As a rich source of manually defined entities, Wikipedia is a good case study for entity ranking. The Initiative for Evaluation of XML Retrieval (INEX) proposes the Entity Ranking track [15] to assess different ranking techniques over Wikipedia. Nevertheless, this is a young field and there is only a handful of research that tackle the Wikipedia entity ranking problem.

Early approaches in entity ranking try to adopt discriminative models in traditional IR, such as using *tf-idf* weights. Vercoustre et al. [16] attempt to exploit results of search engines onto Wikipedia as a special corpus. They represent entity as a document. Given a query in the form of an entity name, they rank documents of Wikipedia to the entity of interest based on three scores:

- rank of the document based on typical search engine (Zettair [3])
- similarity score between the categories of the document and the categories of the entity of interest
- link score of the document, based on the number of links to it from other articles in Wikipedia

[2] Online version is available at http://e-culture.multimedian.nl/demo/session/search
[3] http://www.seg.rmit.edu.au/zettair/

In other approach, Zaragoza et al. [13] try to rank entities by introducing the notion of *entity containment graph* (Figure 2.4). This is a bipartite graph where nodes are entities and excerpts of Wikipedia articles, and edges connect an article excerpt to the entities that appear in the excerpt. Each entity node $e$ is weighed by its *inverse entity frequency*:

$$ief = \log(\frac{N}{n_e})$$

where $N$ is the total number of all Wikipedia article excerpts and $n_e$ is the number of excerpt containing $e$. The rank of entities is the PageRank applied on the entity containment graph with entity weight assigned initially. To facilitate their ranking, Zaragoza et al. build a data set from Wikipedia, which is publicly available for future purposes.

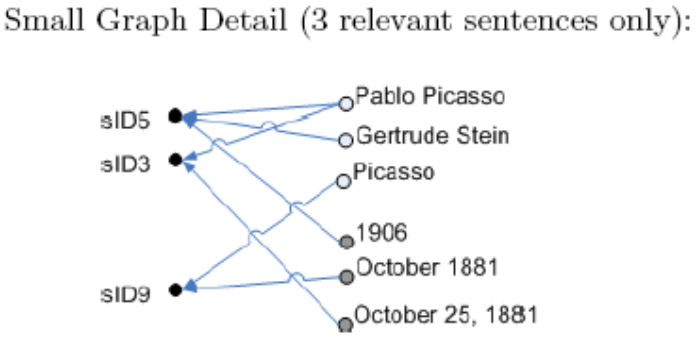


**Figure 2.4:** Example of entity containment graph, from [13]

However, both the approaches by Zaragoza et al. and by Vercoustre et al. do not truly retrieve the entities as the result. While Vercoustre et al. consider documents and entity as interchangeable concepts, and apply traditional IR techniques on a corpus , Zaragoza et al. go to the text snippet level and retrieve passages in response to an entity query. In CATE, we treat entity as the first-class citizen in the knowledge base and return the list of relevant entities instead of any textual representation.

**Language-model-based Ranking.** There is a major trend of adoption the state-of-the-art language models for structured rankings [17, 18]. Language model is a generative model in modern IR, which is based on the probability of generating text features in documents. Chapter 6 explains the language model in more details. In the area of entity ranking, language model is adapted to cope with various formats of documents. Nie et al. [19] extracts a large set of product objects from Web data sources and store in

data records, which they call *Web Objects*. They apply the language model to estimate the probability of generating an object given a set of objects in the database. In [17] and [18], Elbassuoni et al. go further by introducing the language model for a graph of entities and relationships. However, the retrieved results are a sub-graph instead of a list of entities, as in CATE. Furthermore, the authors do not address the context-awareness in rankings.

# Chapter 3

# Data Model

Events in CATE are extracted from two main sources: Wikipedia text corpus and knowledge bases. CATE taps the high quality knowledge bases such as YAGO [2], a huge ontology extracting facts about entities from Wikipedia, to get events with high accuracy. YAGO is a large knowledge base with a rigorous underlying logical model, which is based on Resource Description Framework [20] (RDF) model. In CATE, we extend YAGO to capture the notion of a context, as well as to support retrieval of the textual descriptions of the entities and the relationships between them. This chapter starts with a brief introduction to RDF as a prerequisite; then it describes the YAGO knowledge base and the extensions in CATE.

## 3.1 Resource Description Framework - RDF

RDF was designed to describe concepts and metadata across various resources on the Web. Due to its simplicity, RDF has become a W3C standard for representing knowledge, and for storing and exchanging information in Semantic Web activities. The key point in RDF is that everything (people, cities, artifacts, concepts, etc.) is uniquely represented by a *resource*, or an *entity* from the ontological point of view [21]. Typically, an entity is identified by a *Uniform Resource Identifier* (URI), which is a string following specific syntaxes [22]. For example, `http://www.w3.org/People/Berners-Lee/` is a URI identifying an entity named "Tim Berners-Lee".

Conceptually, an RDF dataset is a set of assertions about entities in the form of *facts* (or *statements* in W3C's specification). Each fact in RDF is a triple of three attributes: a *subject*, a *predicate* and an *object*; and it indicates that there is a binary relationship (represented by the predicate) between the two entities represented by the subject and

the object respectively. For instance, the fact "Berlin is the capital of Germany" can be stated by the triple (for simplicity, the URI forms of entities are omitted):

<subject:Germany predicate:*hasCaptial* object:Berlin>

Similarly, the fact "Gauss was born on 30-04-1777" can be represented by the triple:

<subject:Gauss predicate:*wasBornOnDate* object:"30-04-1777">

**Class, Individual, Literal.** A resource in RDF corresponds to either a named entity (such as Berlin), a relation name (such as *hasCapital* or *wasBornOnDate*), or a literal (such as "30-04-1777"). Similar entities can be grouped into *classes*. For instance, Gauss and Riemann are grouped into the class `person`. A class is also an entity. Named entities which are not classes are called *individuals*. For example, Germany is an individual.

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.

**RDFS.** RDF was extended to *Resource Description Framework Schema* (RDFS ) with new specifications and vocabularies to structure RDF facts in a more expressive way. Among other improvements, RDFS allows multiple classes to be grouped further in a super-class by introducing a new relation name *subClassOf*, or enabling literals to have certain data types (such as Integer, Date, String, etc.). Further explanation of RDFS goes beyond the scope of this thesis. More detailed description can be found in [23].

## 3.2 The YAGO Model

### 3.2.1 Introduction

YAGO is a large ontology with high coverage and precision [24], which is automatically derived from Wikipedia and Wordnet. The first version of YAGO contains more than 1.7 million entities and 15 million facts about them. YAGO has a rigorous underlying logical model and employs type checking techniques to gain the very high precision (approximately 95% for the first version).

As mentioned in Chapter 2, YAGO extracts facts about entities from Wikipedia infoboxes and category pages. An Infobox is a special table-style section in a Wikipedia article and consists of basic information about the entities described in the article. A category page is a special Wikipedia article consisting the list of articles or and sub-categories

grouped under it, as well as of meta-data information such as the meaning of the category. YAGO builds a taxonomy from Wikipedia categories and integrates it with Wordnet [7], benefiting from both the huge number of individual entities in Wikipedia and the accurate taxonomy of concepts in Wordnet. Further details about YAGO can be found in [21].

### 3.2.2 The YAGO Model

**Basics.** YAGO uses the same knowledge representation as RDFS. It identifies concepts and objects in the knowledge base as *entities* with unique names (these names are not identified by the URI, but by the string following a specific syntax [21]). Two entities can be connected with a binary relation through an RDF-style triple as illustrated in Section 3.1. Binary relations have predefined names and semantics, some of them are employed from RDFS vocabulary, some are designed explicitly in YAGO, with the special namespace `yago`. Facts in YAGO are identified by *fact identifiers*, which are also entities themselves (more precisely, fact identifiers are literals of type Integer). Fact identifiers are locally stored in the knowledge base.

More formally, YAGO might be seen as a semantic graph where nodes are entities and labeled edges are relation names. Figure 3.1 shows an excerpt of the YAGO RDF-graph.

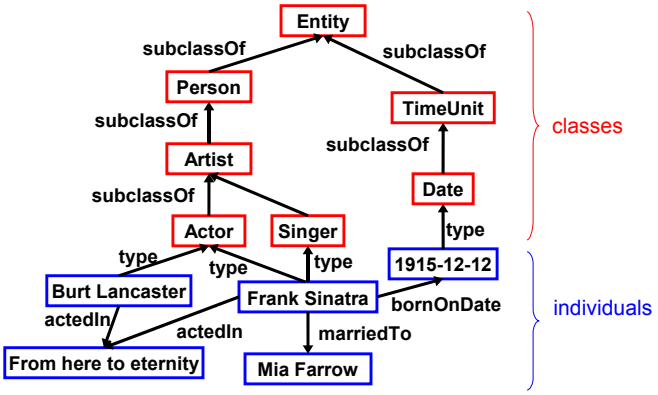


**Figure 3.1:** Excerpt of YAGO RDF Graph, from [3]

**Data type.** YAGO groups literals into a *literal class* or a *data type*. RDFS also has data types, but it uses those defined using the XML Schema (specification in [25]), which does not always cover the semantics of the data type. For example, rational numbers are all numbers that can be expressed as fraction of integers, but in RDFS, there is no way to express them (with existing data types Long, Integer, Decimal). YAGO, on the other hand, defines its own hierarchical system of data types, as illustrated in Figure 3.2

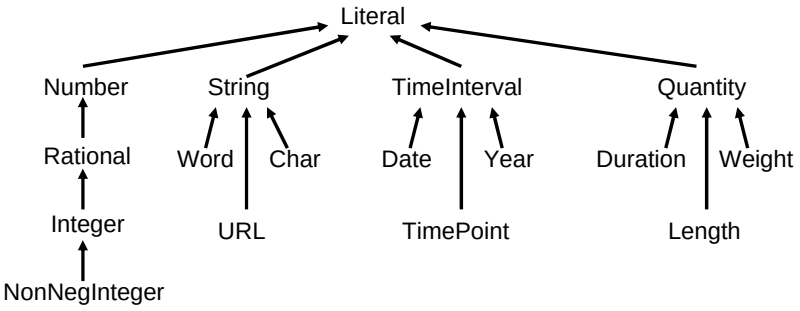


**Figure 3.2:** Data types of YAGO literal, from [21]

### 3.2.3 YAGO2

**Introduction.** YAGO2 [26] [1] has been significantly extended from YAGO by extracting more facts and entities from more resources and with refined methodology. YAGO2 taps Wikipedia, Wordnet and GeoNames to extract more than 10 million entities and more than 80 million facts about these entities. YAGO2 is also confirmed to have a high precision (more than 95% [2]).

**SPOTLX model.** Unlike YAGO model, YAGO2 model slightly deviates from the RDF representation. Facts are still represented as triples subject-predicate-object, but are equipped with additional assertions about time, locations, and contexts. This results in six arguments to represent knowledge: *Subject*, *Predicate*, *Object*, *Time*, *Location* and *ConteXt* (or SPOTLX for short). The *time* argument is a literal of the type TimeInterval. The *location* argument is literal with a special form <longitude, latitude>, where longitude and latitude are of the type Number. The *context* argument is merely a plain string literal indicating some relevant keywords (e.g. anchor texts, Wikipedia categories, etc.) to the fact [3]. Table 3.1 shows some examples of facts of YAGO2.

**Table 3.1:** Example facts in YAGO2, from [26]

| FactId | S | P | O | T | L | X |
|---|---|---|---|---|---|---|
| #1 | GD | *performed* | TCOW | "1978-12-31" | "-37.5,122.3" | "Wall of Sound..." |
| #2 | #1 | *occursIn* | SF | | | |

[1] Hereafter, unless explicitly mentioned, we will use YAGO to refer to both the YAGO knowledge base and the YAGO2 knowledge base

[2] Evaluation results are available on http://www.mpi-inf.mpg.de/yago-naga/yago/evaluation.html

[3] Note that this concept of `context` is different from the concept defined in Section 3.3.1

**Location in YAGO2.** YAGO2 defines a new class named `yagoGeoEntity` to group all entities about locations or areas. For example, Germany is recognized as a location if there exists the following fact in YAGO2:

<Germany *type* `yagoGeoEntity`>

## 3.3 Extension in CATE

CATE uses the same model as YAGO, but extends it with new relation names to capture the notion of context. Beside that, CATE also defines the correspondence between individuals in the knowledge base and documents (Wikipedia articles) in the corpus. As a consequence, it supports the retrieval of entities based on their corresponding documents.

By inheriting YAGO model, CATE also uses the RDFS representation model. All facts are stored as the RDF triples. In CATE, names of entities are carefully designed to match counterpart entities in YAGO. In addition, new relations have been introduced (Section 3.3.2). We see our contribution as the extension of the semantic graph with existing entities in YAGO.

**Entity set.** The input entity of the timeline in CATE must correspond to one individual in the knowledge base, for the system to return a non-empty result. The set of all input entities in CATE forms a subset of individuals set and is called the **Entity set**, denoted by $E$.

### 3.3.1 Context

**Introduction.** A context is a circumstance that forms the setting for an event, statement, or idea, and in terms of which it can be fully understood [1]. In CATE, contexts are identified by three attributes: Time, location and topic. Time indicates the period or the moment in time when the entity is active (for example, the lifetime of a person); location implies the surrounding or relevant places of the entity (for example, the place of birth of a person, or the places where she has spent for a significant amount of time); and topics are the concepts that shed light on various aspects of the entity. For example, the context of Carl Friedrich Gauss should have the following attributes:

- time: 18-th century, 19-th century,..
- location: Braunschweig, Germany,...

[1] Oxford Dictionary

- topic: number theory, differential geometry, astronomy, mathematics,...

meaning that Gauss lived in the 18-th century and 19-th century, was born and spent most of his life in Germany, and served as a mathematician, an astronomer and so on. Note that although a context is defined by three attributes, some times one or more attributes might be missing, for instance the contexts of the entity Analytical Geometry have no time argument. Furthermore, the contexts can have several time interval attributes, as illustrated above in Gauss'case.

**The data model.** From the ontological point of view, a context object is also an entity in the knowledge base. The name of the context follows a specific syntax that distinguishes it from other entities. Context entities are grouped under the class `context`. The attributes of a context are also entities. Time attributes are literals of the type `TimePoint` or `TimeInterval`. Location and topic attributes are named entities. CATE groups location attribute entities into the class `locations`. This class is a sub class of the class `yagoGeoEntity`. Topic attributes are not allowed to match any existing individuals in YAGO, they have to be either an existing class in YAGO or a newly introduced entity. For example, Barack Obama cannot be a topic attribute, since it is a YAGO individual.

**Definition 3.1** (Context)**.** An context is an entity of the type `context`.

For a context $c$, the new triple: $<c, type,$`context`$>$ is added to the YAGO base. Typically, we name a context by concatenating its attributes (e.g., "German_mathematicians"). A set of all contexts are denoted by $C$.

### 3.3.2 Relation Names

#### 3.3.2.1 Context Relation names

We introduce one new relation name to capture the relationships between entities and contexts, and three new relation names for the relationships between a context and its attributes.

**hasContext:** $E \mapsto$ `context` associates one input entity $e$ in the Entity set with a context $c$ by the RDF triple: $< e, hasContext, c >$

**hasTime:** `context` $\mapsto TimeInterval$ associates one context $c$ with a time value $t$ by the RDF triple: $< c, hasTime, t >$

**hasLocation:** `context` $\mapsto$ `locations` associates one context $c$ with a location value $l$ by the RDF triple: $< c, hasLocation, l >$

**hasTopic:** `context` $\mapsto \overline{E}$ associates one context $c$ with a topic value $t$ (t must not be an individual in the Entity set)) by the RDF triple: $< c, hasTopic, l >$

Table 3.2 shows example facts between the context 18th-century_German_mathematicians and the entity Carl Friedrich Gauss, as well as between this context and its attributes:

**Table 3.2:** Example facts about contexts in CATE

| FactId | Subject | Predicate | Object |
|---|---|---|---|
| #1 | Carl Friedrich Gauss | *hasContext* | *c* |
| #2 | *c* | *hasTime* | "'18th-century" |
| #3 | *c* | *hasLocation* | Germany |
| #4 | *c* | *hasTopic* | Mathematicians |

#### 3.3.2.2 Relationships between contexts

The relationships between contexts are naturally specified based on the relationships between their attributes. The relationships between attributes are derived from YAGO by inheriting the hierarchies of concepts (e.g., `mathematics` and `science`) and locations (e.g., `Germany` and `Europe`). To inherit such hierarchies, CATE employs two existing relation names in YAGO, *isLocatedIn* for the relationships between locations, and *subClassOf* for the relationships between topics (examples are in Table 3.3):

**subClassOf:** $\overline{E} \mapsto \overline{E}$ indicates there is a subsumption relationship between two topics. Note that, since a topic is neither an individual nor a literal, if it matches any entity in YAGO, this entity will be a class. Therefore the constraint of the relationship $subClassOf$ in YAGO (which requires both the domain and the range are classes) is still reserved.

**isLocatedIn:** `locations` $\mapsto$ `locations` indicates there is a subsumption relationship between two locations. In YAGO, both the domain and the range of $isLocatedIn$ have the type `yagoGeoEntity`. Our definition is consistent with YAGO model, since the class `locations` is the sub-class of the class `yagoGeoEntity`

### 3.3.3 Entities and Articles

CATE proposes a paradigm for entity retrieval. The key idea is to have a representing document for each entity, and to apply traditional techniques of information retrieval for

**Table 3.3:** Sample context attribute relationships

| FactId | Subject | Predicate | Object |
|---|---|---|---|
| #1 | German_mathematicians | $hasLocation$ | Germany |
| #2 | German_mathematicians | $hasTopic$ | Mathematicians |
| #5 | European_scientists | $hasLocation$ | Europe |
| #6 | European_scientists | $hasTopic$ | Scientists |
| #7 | Mathematicians | $subClassOf$ | Scientists |
| #8 | Germany | $isLocatedIn$ | Europe |

textual documents. In CATE, each entity in the Entity set corresponds to one unique article in Wikipedia (identified by the URL), making this paradigm possible.

**Definition 3.2** (Entity-Article correspondence)**.** A document $D(e)$ is called the article of an entity $e \in E$ if $D(e)$ represents a Wikipedia article that describes $e$.

The correspondence from entities to articles allows inferring the relationships between entities through their documents' hyperlinks. This enables exploiting link analysis algorithms in information retrieval to the entity retrieval problem.

**Definition 3.3** (Entity linkage)**.** An entity $e_1$ is said to be linked to an entity $e_2$ if there is a hyperlink from the article $D(e_1)$ to the article $D(e_2)$. If there is another hyperlink from $D(e_2)$ to $D(e_1)$, then we say $e_1$ and $e_2$ are mutually linked.

### 3.3.4 Event model

Events are actual or contemplated facts of anything happening during the course of time [1]. Events are essential to capture the important information of an entity on the timeline. In CATE, an event is modeled as a group of facts and at least one of them has a time argument. For example, the event "In 1921, Albert Einstein was awarded the Nobel Prize in Physics" can be stated by the facts:

<"#1" Albert Einstein $hasWonPrize$ Nobel Prize in Physics>
<"#2" "#1" $happenedIn$ "1921">

While the facts in an event can have multiple time arguments, only one of them is used as to represent the time of an event. Moreover, each event in CATE can be elaborated with an event description, which is a plain text.

**Definition 3.4** (Event)**.** An event $T$ is a tuple $(t, F, s)$, where $t$ is a literal of type TimePoint or TimeInterval and is called *primary time* of $T$, $F$ is a set of facts and is called factset of $T$, $s$ is a plain literal and is called description of $T$.

[1] Oxford English Dictionary

# Chapter 4

# System Architecture

This chapter gives an overview of CATE's system architecture. As shown in Figure 4.1, CATE consists of five main components: A graphical user interface (GUI) standing in the frontend, a retrieval engine processing users' requests on-the-fly, an event-description extractor to transform results into a user-intuitive format, a data store containing all the necessary data of the system (facts, contexts, relevance relationships between entities, etc.), and an information-extraction tool populating the data store.

The main work-flow of CATE is as follows. The information-extraction (IE) tool extracts data from Wikipedia and the Web and fills the data store (see Section 4.4). It works independently from the other components, and is triggered either periodically by the system or by a personnel. The GUI interacts online with the user by accepting the name of an entity from a text box. The system then passes it to the retrieval engine to get all events as well as contexts relevant to the input entity (see Chapter 6 for more details). These are further sent to the event-description tool which extracts all needed textual descriptions and pictures. Finally, the resulting events and contexts are sent back to the GUI to be output to the user. The user can also interact with the interface by choosing the contexts of interest. In this case, the GUI sends a new request to the retrieval engine. The retrieval engine fetches relevant entities within the chosen contexts and dispatches the results to the event-description tool. Returned events are then updated on the timeline. The following sections will explain all those components in more details.

## 4.1 The Graphical User Interface

The Graphical User Interface (GUI) is responsible for communicating with the user. The user provides the name of an entity, and the GUI returns the relevant events and

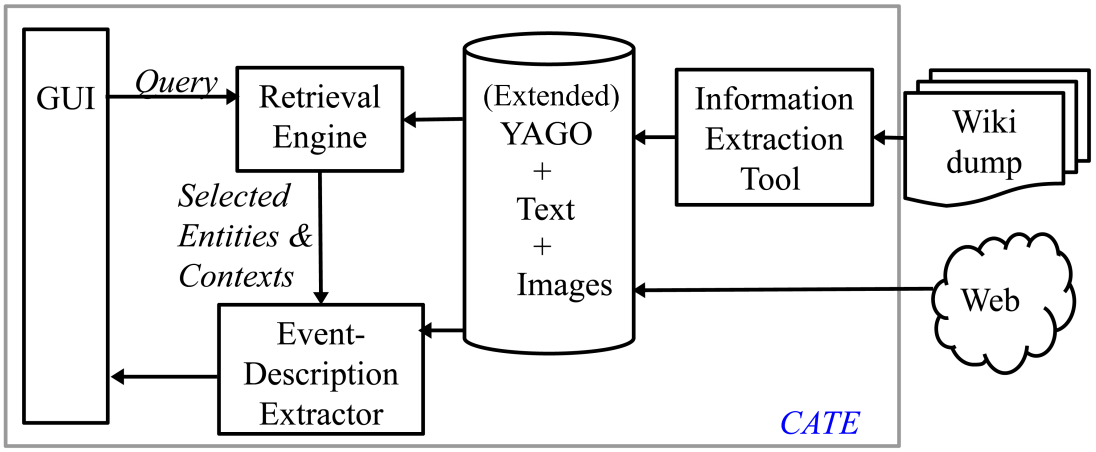


**Figure 4.1:** System Architecture

contexts. The user can choose one or more contexts of the entity, and the GUI responds with the relevant entities to the entity within the chosen contexts. Figure 4.2 shows a snapshot of the GUI when the input entity is Carl Friedrich Gauss.

The GUI consists of two main sub-components. The first sub-component is the *timeline illustrator* where related events are positioned. Each event is displayed with a text snippet and a possible illustrative picture. In addition, on the timeline illustrator, the relevant entities to the input entity are also displayed with text snippets and a description picture (e.g. a portrait of a person). The second sub-component is the *context selector* (the upper part of Figure 4.2), which shows relevant context objects of the entity. We chose not to display contexts directly, but to show their attributes, making it more intuitive to the user. Since contexts' attributes may be overlapping, only distinct values are displayed in the GUI. The user expresses her interest in a certain context by choosing the attributes that best describe it. The system then searches in the knowledge base all the contexts having chosen attribute values. It then uses these contexts to interact to the retrieval engine, issueing a query consisting the contexts and the entity name, and updates the timeline illustrator with the retrieved result.

## 4.2 The Retrieval Engine

The Retrieval Engine accepts requests from the GUI, retrieves necessary data from the data store and processes them in the following tasks:

1. It gets from the knowledge base all facts about the input entity, filters which facts can be displayed on the timeline illustrator of the GUI. These are facts that

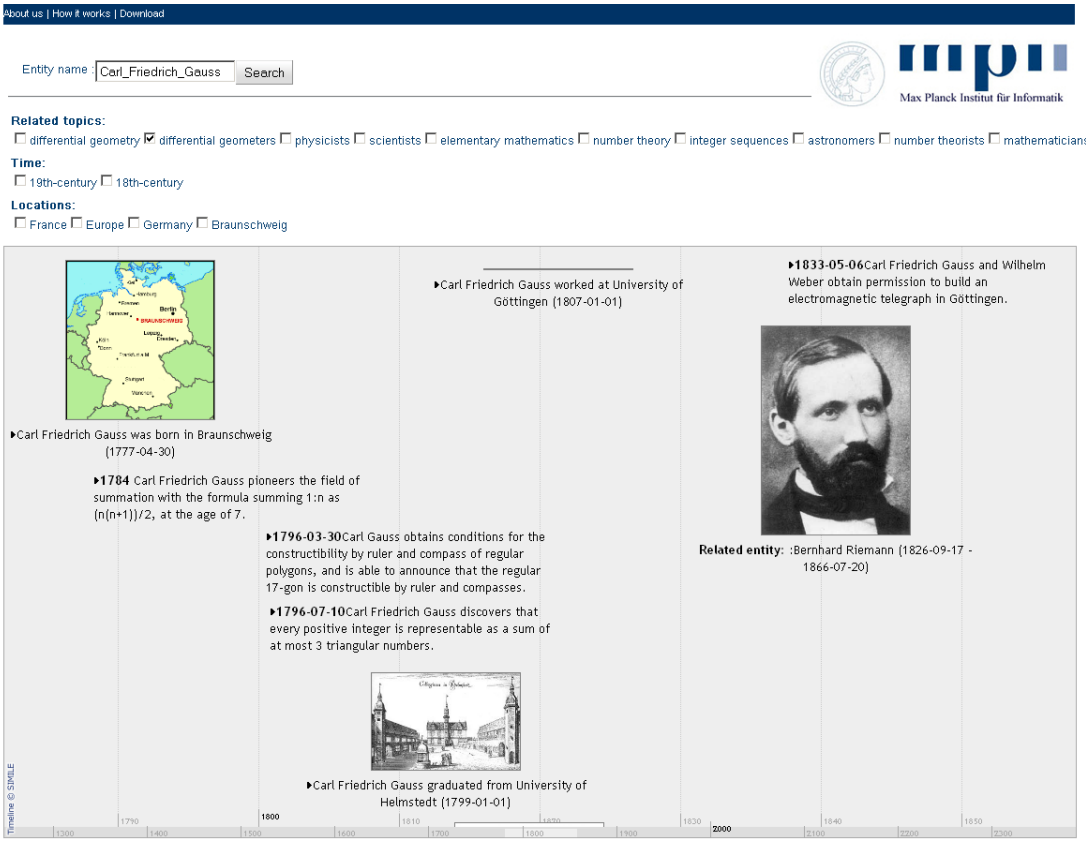


**Figure 4.2:** GUI Snapshot

contain temporal information (e.g., *wasBornOnDate*, *participatedIn*, etc.). All other facts (such as facts with taxonomy relation names, e.g. *type*, *subClassOf*,...) are skipped. It also groups the facts with relation names that semantically describe one single event (e.g., *wasBornOnDate*, *wasBornIn*) into one facts group to build an event.

2. It computes the set of contexts relevant to the input entity, then transforms this set into three sets of time, location and topic values to be used by the context selector.

3. For each chosen context, it retrieves the top-$k$ relevant entities to the given entity and contexts

**Example 1.** The GUI issues a query Carl_Friedrich_Gauss, the retrieval engine gets from the knowledge base the facts such as:

| | | | |
|---|---|---|---|
| "#1": | Gauss | *type* | person |
| "#2": | Gauss | *wasBornOnDate* | "1777-04-30" |
| "#3": | Gauss | *wasBornIn* | Braunschweig |
| "#4": | Gauss | *isCalled* | "Carl Friedrich Gauss" |

and the contexts such as:

| | | |
|---|---|---|
| German_astronomers | $hasLocation$ | Germany |
| German_astronomers | $hasTopic$ | astronomers |
| 18-th_century_mathematicians | $hasTime$ | "17##-##-##" |
| 18-th_century_differential_geometers | $hasTopic$ | Differential_geometers |

The retrieval engine filters facts #2 and #3, and groups them together since they describe one event (Gauss was born in Braunschweig on 30 April, 1777). This group of facts is sent to the Event-Description Extractor (Section 4.3). For the retrieved contexts, the retrieval engine transforms into three sets of time ("17##-##-##"), location (Germany), and topic (astronomers, Differential_geometers).

If the GUI sends requests consisting the context "18-th_century_differential_geometers", the retrieval engine gets from the knowledge base the top-$k$ entities most relevant to Gauss and in the context of $18^{th}$ century mathematicians (Bernhard_Riemann, Jacobi, etc.).

The overall work-flow of the retrieval engine is illustrated in Figure 4.3. The retrieval engine is implemented using a novel entity-ranking model, which is explained in more details in Chapter 6.

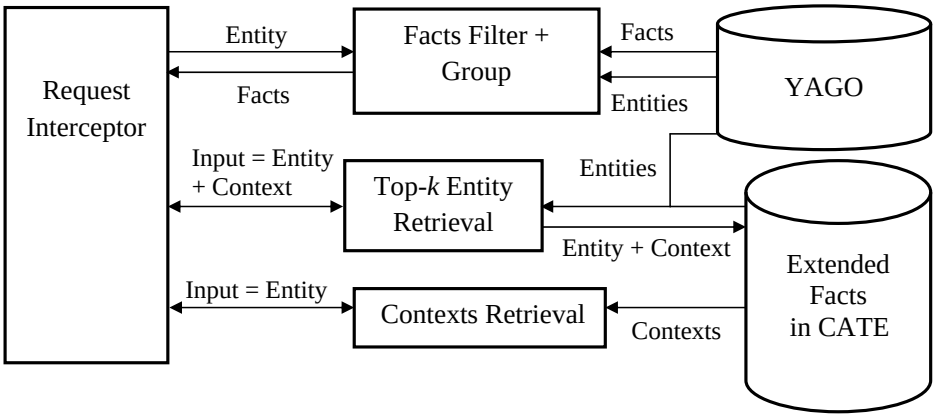


**Figure 4.3:** Retrieval Engine Work-flow

## 4.3 Event-Description Extractor

The event-description extractor is responsible for two main tasks. First, it converts a group of facts received from the retrieval engine, which are represented in RDF triples,

into a human-readable format. Second, it extracts descriptions of another events from the data store.

### 4.3.1 Converting Groups of Facts to Events

As defined in 3.4, an event consists of a group of facts, a text snippet, a possible picture and a timestamp. The timestamp is derived directly from the facts in the group. In example 1, for the group of two facts #2, #3, the extractor will acquire "1777-04-30" as a timestamp.

To extract the text snippet, we integrate results from [27]. The tool takes as an input the corresponding Wikipedia article of the given entity, and a query in which keywords are either an entity name, a context name or both. It issues the query to the article and retrieves the most relevant text fragments to the keywords.

Finally, to obtain the illustrative pictures, we use the existing tool [28]. For a given entity name, it makes use of a Web search engine to extract the relevant pictures to the entity.

### 4.3.2 Extracting Additional Events

Events constructed from the groups of facts sent from the retrieval engine only cover a small proportion of events of the entity. In CATE, we develop an extraction framework which parses the Wikipedia article of the entity, as well as other Wikipedia articles having links to the entity, and extracts all text snippets and timestamp that describes an event of the entity. The results are stored in the event data store. The event-description extractor then retrieves them to output to the GUI as additional events. We discuss this in more details in Chapter 7.

## 4.4 Information-Extraction Tool

The Information-Extraction (IE) tool in CATE is used to extract data from Wikipedia and to extend the YAGO knowledge base with facts about contexts, as mentioned in Section 3.3. Additionally, the IE tool extracts from Wikipedia various information from Wikipedia such as full text, hyperlinks of entities, which are used by both the retrieval engine and the event-description extractor. The extraction results are unified with existing entities in YAGO. We explain in more details our extraction algorithms in Chapter 5.

## 4.5 Data Store

The data store in CATE contains three main databases. The first database is a RDF database, which consists of a portion of the YAGO knowledge base extended with new relationships and entities described in Section 3.3. More specifically, this database gets from YAGO all entities that have a corresponding article in Wikipedia. For each such entity, the database stores all contexts associated with it, and the attributes associated with the contexts.

The second database is the text database. It stores information about articles of entities derived from the Wikipedia dump [1], which is publicly available on the Internet. We construct from the Wikipedia dump two different indices for our purpose:

1. *Hyperlink index.* This is an inverted index over the entity links. In particular, for each entity, it stores all the Wikipedia articles that have outgoing links to it and the number of such links present in each article.

2. *Full-text index.* This stores for each entity the full text of its article. The full text is pre-processed with existing tools such as [29] and is stored in XML format with rich meta data such as length (in links and in words) of the article, or list of Wikipedia categories that the article belongs to. This information is used by the IE tool, the retrieval engine as well as the event-description extractor.

The third database is the image database, extracting images using [28]. It contains an image for each individual entity in YAGO. By storing images in a local database, the response time of CATE can be reduced significantly each time an entity is queried.

[1] http://en.wikipedia.org/wiki/Wikipedia:Database_download

# Chapter 5

# Information Extraction

In this chapter, we discuss the algorithms and the techniques we develop for extracting new structured data from Wikipedia to extend the YAGO knowledge base, and to populate our data store. This involves three main tasks: 1) Extracting facts about contexts (their attributes, their relationships with entities); 2) Extracting text from Wikipedia (article text and hyperlinks, description of events); 3) Extracting images for entities from the Web. This chapter focuses on the first task, the other two will be discussed further in Chapter 7.

Extracting facts about contexts entails three sub-tasks: 1) Extracting attribute vocabularies, 2) constructing context objects with its attributes, and 3) constructing the relationships *hasContext* between contexts and entities. For extracting attribute vocabularies, we use Wikipedia category titles and the hierarchy of categories. The intuition is that, Wikipedia enables users to group similar articles to one category, and to group similar categories to one super category. Hence, the Wikipedia category hierarchy reflects users' perspective about the contextual background of articles. Moreover, category names are standardized according to the Wikipedia's thorough editing guidelines [1]. Category names are often composite names such as "18th_century_mathematicians" or "German_mathematicians", in which constituent elements (e.g., "18-th_century" or "mathematicians") correspond to time, location and topic properties. This observation forms a basis for our baseline extraction algorithm (Section 5.1.

For constructing context objects, we also use Wikipedia categories. We map each category to at most one context object. However, since Wikipedia categories are noisy (e.g., some do not contain any articles, some are used for administration purpose only, etc.), not all Wikipedia categories are mapped to one context object. We maintain the set of

[1] http://en.wikipedia.org/wiki/Wikipedia:Categorization, http://en.wikipedia.org/wiki/Wikipedia:Naming_conventions_(categories)

rules to filter ill-formed Wikipedia categories. These categories will not be mapped to any context. Moreover, we expand the set of contexts by combining different attribute vocabularies. In other words, context objects are extracted from a part of Wikipedia categories, and from attributes combination. We explain this in Sections 8.3 and 5.3.

To associate entity to contexts via the relation name *hasContext*, we use two strategies, which are based on categories of the corresponding article, and on link analysis. We discuss this in more details in Section 5.3.

## 5.1 Extracting Attribute Vocabularies

### 5.1.1 Time Value Extraction

Time values are extracted directly from Wikipedia categories through pattern-based rules. These rules recognize all category names matching a particular pattern and output time values as literals of the format "yyyy-mm-dd" (ISO-8601 [30]). For instance, the following rule will detect and output the time value "17## − ## − ##" from the category 18-th_century_mathematicians:

*Rule* #1: *pattern*:[*Y*-th_century_[TOKENS]] *output*:"[*Y*-1]##-##-##"

In the above example, "17## − ## − ##" is the representation of 18-th_century in ISO-8601 formats, where "#" is the wildcard symbol for unknown information.

### 5.1.2 Location and Topic Values Extraction

While time attributes are literals, location and topic attributes are named entities (Section 3.3.1). Existing pattern-based approaches (e.g., [8]) cannot be easily applied in extracting locations and topics, since the results are affected by the ambiguity in entity names (for example, America can be a country, or a musical band in London).

In CATE, we extract topics and locations in two steps. In the first step, we generate the patterns automatically using a *pattern learning algorithm* (Algorithm 5.1). In the second step, we use the patterns to extract candidate names for locations and topics (Section 5.1.2.3). These candidates are then mapped to named entities in YAGO using some entity mapping techniques (Section 5.1.3).

#### 5.1.2.1 Preliminaries

Before explaining the algorithm for learning patterns, we need to introduce the notations and concepts below.

**Token.** In Wikipedia, all article names have the standard format X_Y_Z, where X, Y, Z are strings without underscores. These strings are called *tokens*. The tokens between parentheses or the double quotes can be grouped into one composite token. For example, "Jim_Gray_(Computer_scientist)" has three tokens: Jim, Gray, (Computer_scientist).

**Positioned token.** A positioned token is a token associated with its positions in the category. For example, "Jim_Gray_(Computer_scientist)" has three positioned tokens: <Jim, 1>, <Gray, 2>, <(Computer_scientist), 3>.

**Pattern.** A pattern is a set of positioned tokens. For example, the pattern "Lists_of_XX_from_Y", where Y indicates a single token and XX is either one or two tokens, correspond to a set $S = \{<$ Lists,1>,< of,2>,< from,{4,5}>$\}$.The positioned <from, $\{4, 5\}>$ means the token from can appear in the 4-th place (if XX is a single token) or 5-th place (if XX is two tokens) in the category.

**Definition 5.1** (Pattern confidence)**.** For a set $W = \{w_1, w_2, ...\}$ of Wikipedia categories and a pattern $P$, we measure the *confidence* of $P$ in $W$ as:

$$conf(P, W) = \frac{|W|_P}{|W|} \tag{5.1}$$

where $|W|_P$ is the number of $w_i$ matching pattern $P$, and $|W|$ is the total number of categories in $W$. When $conf(P, W)$ is equal to 1, this means that the pattern $P$ perfectly describes the set $W$. When $conf(P, W)$ is equal to 0, this means that $P$ is not applicable in $W$.

#### 5.1.2.2 Pattern Learning Algorithm

As a bootstrapping step, we use the Wikipedia categories that follow the pattern [X_by_Y]. This type of category indicates the partitioning of the concept X into sub-categories (or articles) where the elements share the same value of the attribute Y. For instance, "Mathematicians_by_country" groups all sub-categories (or articles) about mathematicians, each of them group a list of articles (or other sub-categories) about mathematicians in a specific country (Figure 5.1). The names of these sub-categories (e.g., Hungary, Germany,

Japan) reveal instances of the concept country. Hence, in order to extract values of location attributes, we can focus solely on the categories of the pattern [$X$_by_$Y$], where $Y$ is country, city, etc. (Table 5.1)

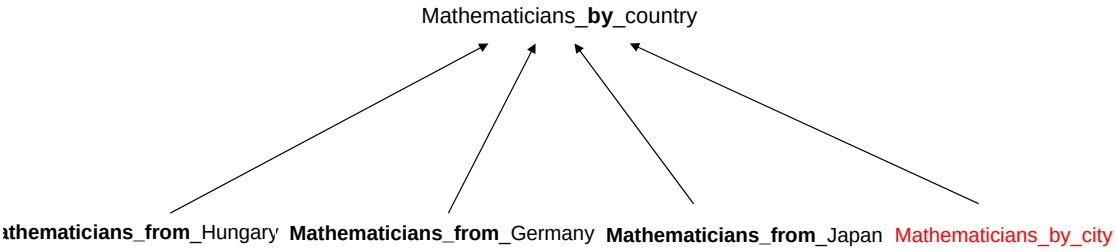


**Figure 5.1:** Example of a category with the pattern [X_by_Y]

Our algorithm takes as an input the set $W$ of categories. In our case, $W$ represents a set of sub-categories of a Wikipedia category of the pattern [X_by_Y]. The algorithm outputs the pattern describing $W$ with the highest confidence. Note that this algorithm does not depend on the pattern [X_by_Y]; it can be adapted to any arbitrary set of Wikipedia categories.

**Algorithm 5.1** Learning patterns from Wikipedia categories

**Input:** Set $W$ of Wikipedia categories
**Output:** Pattern $P$ that matches $W$ with highest confidence.

1: For each $w_i \in W$, tokenize $w_i$ into set positioned tokens
2: Merge all positioned token sets into one single set $S$
3: Sort positioned tokens in $S$ by their occurrences in all categories in $W$ in descent order
4: Initialize the pattern $P$ as an empty set: $P = \varnothing$
5: **while** there exists a positioned token $< t, p >$ which is not processed **do**
6: Put $P \leftarrow < t, p >$
7: Compute $conf(P, W)$ and compare with previous value (without $< t, p >$)
8: If $conf(P, W)$ degrades, exclude $< t, p >$ from $P$ and STOP
9: **end while**
10: output $P$

#### 5.1.2.3 Extracting candidates for locations and topics

The work-flow for extracting candidates for locations and topics is shown in Figure 5.2. First, we determine different patterns of form [X_by_Y] with different values of $Y$ as the seeds, and manually define the mapping from these seeds to an attribute (Table 5.1). For each seed, we obtain all matching categories and the set of their sub-categories. The learning algorithm is applied to each set to output the pattern. This pattern is then examined against the sub-category sets to extract the remaining tokens as candidates.

For the example, in Figure 5.1, if the system learns the pattern [Mathematicians_from_Z], then sub-categories such as "Mathematicians_from_Germany" will be parsed to extract the remaining token ("Germany") as a candidate for location attribute.

**Table 5.1:** Different pattern [X_by_Y] and mappings $Y$ to context dimensions

| Id | Pattern | Mapping of $Y$ |
|---|---|---|
| 1 | X_by_country | locations |
| 2 | X_by_nationality | locations |
| 3 | X_by_city | locations |
| 4 | X_by_region | locations |
| 5 | X_by_province | locations |
| 6 | X_by_field | topics |
| 7 | X_by_field_of_research | topics |
| 8 | X_by_field_of_study | topics |
| 9 | X_by_occupation | topics |
| 10 | X_by_subject | topics |
| 11 | X_by_specialty | topics |

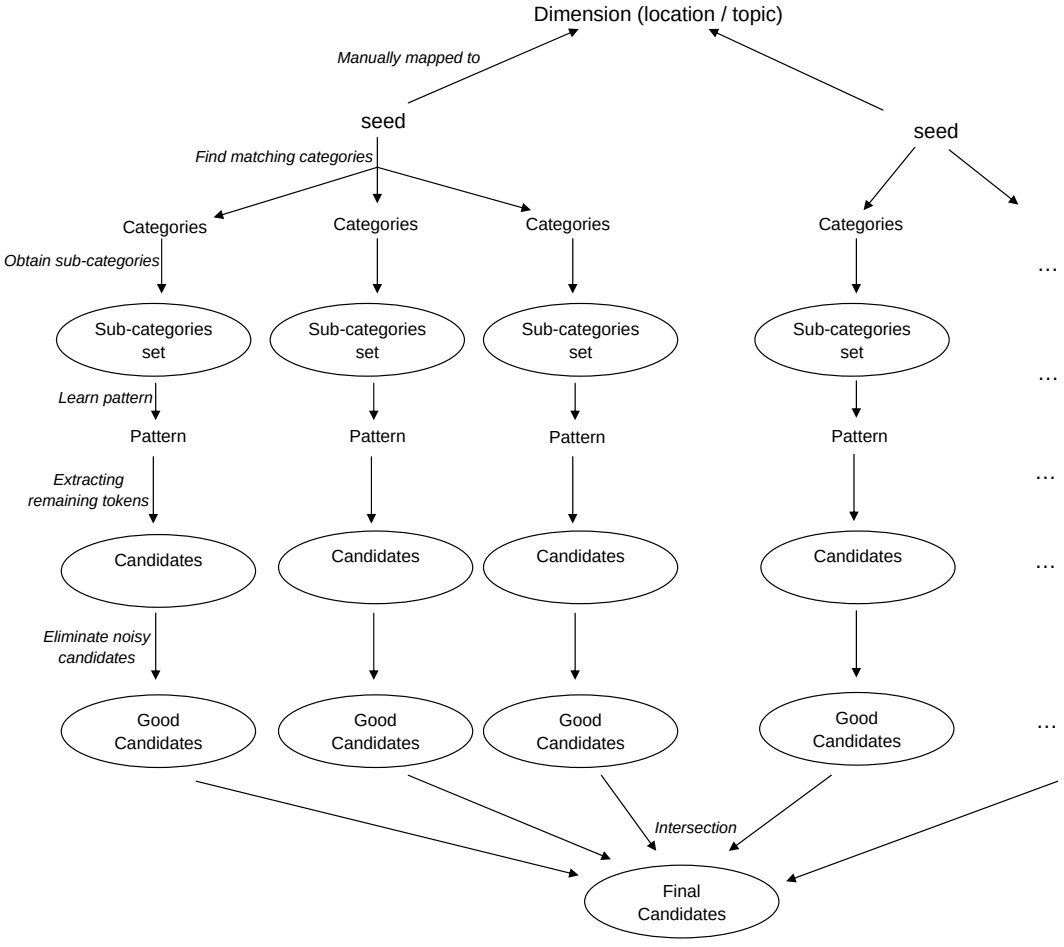


**Figure 5.2:** Candidates Extraction Work-flow

In practice, candidates extracted by the above procedure are not always precise. For example, from the set of sub-categories of List_of_people_by_occupation, CATE learns the pattern [Lists_of_XX] and extracts good candidates ("astronomers", "'foolbal_players") from categories such as Lists_of_astronomers, Lists_of_football_players, but also noisy candidates ("female_people_by_occupation") from Lists_of_female_people_by_occupation. To improve the quality of candidates, we use a simple heuristics: Candidates that contain extremely popular tokens in Wikipedia (such as "Lists", "stubs", "people", etc.) are eliminated. These popular tokens are obtained by an independent process that crawls and tokenizes all the Wikipedia category names, then counts the occurrences of the tokens over the entire corpus.

After all candidates are extracted for each individual set, we take the intersection of all sets corresponding to a single attribute. Then we remove the very sparse candidates (i.e., candidates which are extracted only in one or two sets). This results in a final set of candidates for topic or location attribute vocabularies. Figure 5.1 illustrates this procedure.

### 5.1.3 YAGO Entity Mapping

Candidates for topics and locations obtained from the previous step are merely plain strings, e.g. "differential geometry" or "the United States of America". To map them to proper named entities in YAGO, we use a simple approximate string matching algorithm based on Levenshtein distance [31], and combine it with type checking techniques to improve the accuracy of the results. We explain our mapping algorithms as follows.

**Mapping location values to YAGO entities.** Each location candidate is run against the YAGO knowledge base to find all matching (or most similar) entity names. If one of these entities has the type yagoGeoEntity, the candidate will be automatically mapped to it. Otherwise, we manually assess if the candidate actually represents a location name, and remove if it does not. Figure 5.3 illustrates the YAGO location mapping in SPARQL language [32], where `levenshtein` is a function for measuring the Levenshtein distance between two strings.

```
PREFIX y :  <http://mpii.de/yago/resource/>
SELECT ?entity WHERE
{    ?entity y:type 'yagoGeoEntity'.
     FILTER levenshtein(?entity, X, 1)
}
```

**Figure 5.3:** SPARQL statement for mapping location candidate X into a YAGO entity

**Mapping topic values to YAGO entities.** The mapping of the topic candidates is performed in the same way, except that when no matching entity in YAGO is found, we check if there is any Wikipedia category has the same name as the candidate. If such category are found, we generate a new vocabulary for the candidate and add it to the knowledge base.

## 5.2 Constructing Context Objects from Wikipedia Categories

To construct context objects from Wikipedia categories, we first preprocess the Wikipedia categories hierarchy to exclude ill-formed Wikipedia categories, for instance skipping stub categories[2] (categories containing articles that need improvement) and maintenance categories [3] (categories used for maintenance purposes and that are not part of the encyclopedia). After that, we parse the name of each Wikipedia category, compute the values of three attributes of the category using the string matching techniques. For example, for the category "18-th_century_mathematicians", we extract the time attribute which is equal to "18##-##-##" and topic attribute which is equal to Mathematicians.

In fact, this task is not as easy as it first seems. One of the reason is that string matching does not recognize different representations of the same concept. For example, given the category "Chinese_philosophers", it does not recognize the location People's_Republic_of_China. In CATE, we adopt some heuristics to tackle this issue. First, we always prefer values with longer name when looking for attribute presence in a category name. Second, we employ an existing linguistic tool [33] to acquire as many variants as possible for a given word. For example, "United_States_of_America" has variants "United_States", "the_United_States", "the_United_States_of_America", "American", "America".

## 5.3 Assigning Entities to Contexts

### 5.3.1 Assigning Entities to Contexts Based on Categories

So far we have explained how to extract attribute vocabularies and construct context objects from Wikipedia categories. In CATE, we assign an entity to contexts by adding new facts with the relation name *hasContext* to YAGO. The main source for such facts is the links between categories and the articles they group. Those links are helpful since they exploit the user's effort of grouping entities with the most explanatory categories.

[2] http://en.wikipedia.org/wiki/Category:Stub_categories
[3] http://en.wikipedia.org/wiki/Category:Wikipedia_maintenance

For each entity $e$, we parse the categories of the representing article, and find the corresponding contexts for each category as extracted from Section 5.3.3. Each context is then associated with the entity by one fact with the relation name *hasContext*. For example, to assign contexts for Gauss, we retrieve all categories of the article "Carl Fredrich Gauss". Each category, e.g. "18-th_century_mathematicians", is checked against the knowledge base to find the corresponding context $c$. Then we add new fact: <Gauss, *hasContext*, $c$> into the knowledge base.

### 5.3.2 Assigning Entities to Contexts Based on Links

Wikipedia categories are not enough to assign entities to contexts, they are incomplete and imprecise in many cases. The reason is that, category pages are less visible than articles. Hence, they receive much less scrutiny from the user when generating or editing an article. In addition, there is no mechanism to exhibit the user the full category taxonomy and the semantics of the categories [34], which results in creating redundant categories.

In CATE, we assign an entity to additional contexts by analyzing the entity links and initial context assignment from Wikipedia categories. The idea is that we assign an entity $e$ to the context $c$ if the majority of entities that $e$ links to, or the majority of entities that link to $e$ belong to $c$ as well. For example, in Wikipedia, the category "Modular arithmetic" is not assigned to the article "Carl Friedrich Gauss". However, since most of the articles grouped under this category have links to the article about Gauss, we can assign Gauss to the corresponding context of "Modular arithmetic".

More formally, for each entity $e$ and context $c$, we measure the confidence of assigning $e$ to $c$'s as follows:

$$conf(e,c) = \alpha \frac{|e'|_{c,e}}{|e'|_c} + (1-\alpha)\frac{|e'|_{c,e}}{|e'|_e} \tag{5.2}$$

where $|e'|_{c,e}$ is the number of entities $e'$ which belong to the context $c$ and are mutually linked to $e$ (Definition 3.3), $|e'|_c$ is the total number of entities $e'$ in context $c$, and $|e'|_e$ is the total number of entities mutually linked to $e$.

The confidence scores are computed for each context object $c$ and entity $e$ in the knowledge base, and $e$ is assigned to $c$ if $conf(e,c)$ is greater than a predefined threshold.

### 5.3.3 Constructing Context Objects From Attributes

The assignment of entities to contexts based on categories and on hyperlinks would only find contexts that are part of Wikipedia categories. However, new context objects can be generated by joining attributes from different contexts of the same entity. For example, Gauss belongs to the context 18th_century_mathematicians and German_mathematicians, and we can generate a new context object 18th_century_Germany. This strategy can go further by utilizing external knowledge sources about location or topic taxonomy to generalize attribute values, and thus provide wider context objects. For example, given the fact that Germany is part of Europe, the context 18th_century_Germany can be generalized to a new context 18th_century_Europe.

In CATE, we construct new context objects and at the same time assign them to existing entities. Given an entity (e.g., Gausss), for contexts that are already associated with the entity, we combine their attributes of different type time, location and topic to generate new context objects. We also expand location vocabularies by exploiting the facts with relation name *isLocatedIn* from YAGO. For example, given the location Germany, we obtain from YAGO the following fact:

<Germany $isLocatedIn$ Europe>

then a new location attribute Europe is included to existing contexts to generate a new context object, for example from German_mathematicians we have Europe_mathematicians. Finally, the new context objects is associated with the entity with a fact with the relation name $hasContext$, for example:

<Gauss $hasContext$ Europe_mathematicians>

## 5.4 Summary

In this chapter we discuss algorithms and techniques we use in our information extraction component. This component extracts from Wikipedia the facts about contexts and attributes with three relation names: *hasTime*, *hasLocation*, *hasTopic*; and extracts the relationships between entities and contexts by constructing facts with the relation name *hasContext*. All these facts are stored in our extended knowledge base. In the next chapter, we will discuss how we retrieve entities from our data store using the new ranking model.

# Chapter 6

# Entity Ranking Model

In this chapter, we describe our model for ranking the contexts and the entities relevant to the entity of interest. We have three intermingling ranking problems: 1) ranking the contexts given an entity, 2) ranking the entities given a context and an entity, and 3) ranking the entities given a certain context only. For all three problems, we adopt a statistical-language-modeling approach [1], and we utilize our text database with its indices to estimate the parameters of our model. We start with a brief introduction to statistical language models in traditional information retrieval, then apply it to define a language model for contexts and entities, and use this model in the retrieval engine of CATE to handle each ranking problem separately.

## 6.1 Statistical Language Model in IR

### 6.1.1 Introduction

Generally speaking, a statistical language model (LM) is a function that assigns a probability to a sequence of words $P(w_1, w_1, \ldots w_n)$ [1]. Such sequences can be phrases or sentences, and their probabilities can be estimated from a large corpus of documents by means of a probability distribution [1]. LM provides a new way to capture users' information need, and has gained increasing attention recently in the IR community [35]. The basic intuition is that when users choose good terms for a query, they often think of keywords that are likely to appear in a relevant document [36, Ch. 12, p. 218]. If each document in the corpus is associated with a language model, and if the query is seen as a sequence of words, then the matching score of the document and the query can be represented by the probability of generating the query based on the corresponding

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

language model. In the next section, we illustrate the idea of language modeling with one of the basic methods, the *query likelihood model* [2].

### 6.1.2 Query Likelihood Model

In the query likelihood model, each document $d$ in the corpus is assigned a LM, called a *document model* of $d$, denoted by $M_d$. The objective is to rank $d$ a given query $q$ by $P(d|q)$, which is the probability of $d$ being relevant to $q$. Using Bayes rule, we have:

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)} \tag{6.1}$$

$P(q)$ is independent of the documents and can be ignored. $P(d)$ is often treated uniformly over all documents in the corpus and thus can also be ignored. The equation in 6.1 is therefore equivalent to:

$$P(d|q) \propto P(q|d) \equiv P(q|M_d) \tag{6.2}$$

In other words, the relevance of a document to a query is equivalent to the probability of the query being generated from the document model, assuming the generation is a random process. In practice, the probability $P(q|M_d)$ is often estimated using a Maximum Likelihood Estimator (MLE) [37, Chap. 9] with unigram assumption:

$$P(q|M_d) = \hat{P}_{mle}(q|M_d) = \prod_{t \in q} \hat{P}_{mle}(t|M_d) = \prod_{t \in q} \frac{tf_{t,d}}{L_d} \tag{6.3}$$

where $tf_{t,d}$ is the term frequency of the term $t$ in the document $d$, and $L_d$ is the number of terms in $d$.

### 6.1.3 Smoothing

One of the issue with the LM is that the occurrences of terms in documents are very sparse. Some words never appear in the document (and thus the LME for the probability of these terms will be zero according to equation 6.3), but are good terms to capture the user's information need. If these words are included in a query, the score $P(q|M_d)$ of a document $d$ will be zero, and the document will be completely irrelevant to the document. This phenomenon is called *overfitting* and is a well-known phenomenon in generative models. The traditional solution to this issue is adopting a *smoothing* technique. This technique adds a small mass to the probability distribution and renormalizes it to make non-occurring terms have a very low, but non-zero probability. This probability

[2] There are also other IR models on top of the statistical language model, for example *document likelihood model* or *Kullback-Leibler divergence model* [36, Ch. 12, p. 230 - 232]

is interpreted as the chance of the term occurring in the document in general. It is somewhat close to, but not greater than the occurrence of the term over the entire corpus (So if the term does not appear anywhere in the corpus, we can safely exclude it from the document model). In other words, the upper bound of a smoothing is [36]:

$$\hat{P}_{mle}(t|M_d) \leq \frac{cf_t}{T} \qquad \forall tf_{t,d} = 0 \tag{6.4}$$

where $cf_t$ is the occurrence of the term $t$ over the whole corpus, and $T$ is the total number of all terms in the corpus.

One of the most common smoothing technique is *Dirichlet smoothing* [38]. It mixes the MLE of the document model with the frequency of the word in the corpus. The probability $P(t|M_d)$ in equation 6.3 is then changed to:

$$P(t|M_d) = \lambda \hat{P}_{mle}(t|M_d) + (1-\lambda)\hat{P}_{mle}(t|M_E) \tag{6.5}$$

where $0 < \lambda < 1$ is the smoothing parameter, and $M_E$ is the probability of a term t being generated from the whole corpus. A good $\lambda$ is crucial to avoid overfitting and make LMs useful. The parameter should ideally be learnt from query logs for instance.

## 6.2 Statistical Language Model for Entity Retrieval

### 6.2.1 Language Model for an Entity

We employ the above language model to solve our three ranking problems. The key point is the introduction of language models for an entity and for a context, which are based on the language model for a document. In particular, we map each entity $e$ to a document $D(e)$, which is the Wikipedia article that describes $e$ (Definition 3.2), and construct a statistical language model for this article. The LM for $e$ is inferred through the LM for the document $D(e)$.

Similarly to the traditional statistical language modeling, we define a statistical language model for an entity $e$ as a function that assigns a probability to a sequence of entities $e_1, e_2, \ldots e_n$ from the knowledge base. This probability is measured by the probability of generating the entity $e_i$ given the LM of the document $D(e)$. Using query likelihood model and the Dirichlet smoothing (equation 6.5), this probability becomes:

$$P(e_i|M_{D(e)}) = \lambda \hat{P}_{mle}(e_i|M_{D(e)}) + (1-\lambda)\hat{P}_{mle}(e_i|M_E) \tag{6.6}$$

where $\hat{P}_{mle}(e_i|M_{D(e)}$ is the MLE of the probability of generating the entity $e_i$ from the document model $M_{D(e)}$, and $M_E$ is the LM trained from all documents of all entities in the knowledge base (or all Wikipedia articles in the corpus). Note that by adopting the equation 6.5, we already assume the generation of entities are independent processes (unigram assumption), i.e. the presence of one entity in a Wikipedia article does not depend on the presence of any other entity in the same article.

There are many ways to estimate $\hat{P}_{mle}(e_i|M_{D(e)}$. In CATE, we exploit the links between entities (Definition 3.3) and adjust equation 6.3. The "term frequency" of an entity $e_i$ in the document $D(e)$ is measured by the number of entities in $D(e)$ that are linked to $e_i$ (equation 6.7). This is based on the observation that having a link to an entity in some Wikipedia article is a much stronger evidence that this entity is relevant to that article as opposed to just being mentioned in the text. Formally, we have:

$$\hat{P}_{mle}(e_i|M_{D(e)}) = \frac{lc(e_i, D(e))}{|D(e)|} \tag{6.7}$$

where $lc(e_i, D(e))$ is the number of links to $e_i$ in the document $D(e)$ and $|D(e)|$ is the length of document $D(e)$ (i.e., the sum of all links to entities in $D(e)$). The Dirichlet smoothing part in equation 6.6 is treated in the same way. Finally, we have:

$$P(e_i|M_{D(e)}) = \lambda \frac{lc(e_i, D(e))}{|D(e)|} + (1-\lambda)\frac{lc(e_i, Col)}{|Col|} \tag{6.8}$$

where $Col$ refers to the whole collection of documents and $lc(e, Col)$ and $|Col|$ is the total number of links to entity $e$ and the total number of links to any entity in the whole knowledge base respectively.

### 6.2.2 Language Model for a Context

We adopt the following notation. Each context $c$ is associated with a set $E(c) = \{e_1, e_2, ..., e_n\}$ which consists of entities that belong to the context $c$. For entity $e_i$, we construct a LM of the document $D(e_i)$ as described in Section 6.2.1. These LMs map each entity $e$ to a probability $P(e_i|M_{D(e)})$ computed by equation 6.8. The language-model $M_c$ for a context $c$ is constructed as a mixture model over the LM of its entities. It maps each entity $e$ to a probability $P(e|M_c)$, which is an aggregation of all $P(e_i|M_{D(e)})$:

$$P(e|M_c) = \frac{1}{n}\sum_{i=1}^{n} P(e|M_{D(e_i)}) \tag{6.9}$$

## 6.3 Entity Ranking Model

We have explained all the components of our ranking model, now we describe how this model is used to solve our three ranking problems stated in the beginning of this chapter.

### 6.3.1 Ranking Contexts

Given an entity $e$ and the set of contexts retrieved from extraction algorithms explained in Chapter 5, we rank the contexts based on their relevance to $e$ and retrieve the top-k results. For each context $c$, we denote its relevance score to $e$ by $P(c|e)$. Applying Bayes rule, we have:

$$P(c|e) = \frac{P(e|c)P(c)}{P(e)} \tag{6.10}$$

We ignore the determinator $P(e)$ since it is independent of the context and does not affect the ranking. The probability $P(c)$ is set uniformly across all contexts in the knowledge base and thus is ignored as well. The ranking of contexts now solely on $P(e|c)$, which is the probability of generating the entity $e$ given the context $c$. Using the LM of the context $c$, we have the ranking of the context $c$ is computed by:

$$P(c|e) \propto P(e|c) \equiv P(e|M_c) = \frac{1}{n} \sum_{e_i \in c} P(e|M_{D(e_i)}) \tag{6.11}$$

where $e_i$ is a context that belongs to the context $c$.

### 6.3.2 Ranking Entities Relevant to a Given Entity within a Context

The second ranking problem we have is to retrieve the most relevant entities given an entity $e$ and a context $c$. We rank the entities based on their probability of being generated given $e$ and $c$ which we denote by $P(e'|c, e)$. To compute such probability, we also construct a language model for $c$ as a mixture model of the documents LMs of all entities in $c$. However, we only strict this to documents that *contain* $e$ to accommodate for the conditioning over $e$. That is, we ignore the documents of entities that belong to $c$ and do not contain $e$. To this end, let the set of documents of entities that belong to $c$ and contain $e$ be $\{D(e_1), D(e_2), ..., D(e_l)\}$. This way, the probability $P(e'|c, e)$ is equal to:

$$P(e'|c, e) \equiv P(e'|M_{c,e}) = \frac{1}{l} \sum_{i=1}^{l} P(e'|M_{D(e_i)}) \tag{6.12}$$

Again, we estimate $P(e'|M_{D(e_i)})$ using a maximum-likelihood estimator as described above.

### 6.3.3 Ranking Entities within a Context

The third and final ranking problem we deal with is ranking entities $e'$ that belong to a certain context $C$. This can be easily done using the probability $P(e'|M_c)$ which is computed as described in equation 6.9.

# Chapter 7

# Implementation

This chapter explains the implementation of the different components of CATE. As stated in Chapter 4, CATE consists of two modules running independently. The first module, which runs online, accepts requests from the user, then connects to the databases underneath to get the results for the requests that it displays on the GUI. The second module runs offline and it extracts periodically the information from the Wikipedia dump to populate the data store. We first explain how the online module works, then we describe the implementation of the offline module and the data store.

## 7.1 CATE Client-server Framework

We implement the online module of CATE as a Web application using three-tier architecture (Figure 7.1). The client side is responsible for wrapping input data as a Javascript object and generates an HTTP request to the web server, and for rendering the result on the timeline using Javascript and DHTML. The web server is responsible for transforming the request into Java objects and send to the application server, as well as for converting the results and transferring them back to the client. The application server processes all the requests and return the results in a raw format, which are converted into a human-readable format before being sent back to the user. This 3-tier architecture is essential to make CATE flexible and scalable: the web-server layer makes the algorithmic processes in the application server transparent to the client. When the GUI is re-designed, the system can be adjusted with minimal effort. Similarly, when the system is re-implemented using new algorithms, it does not affect the functionalities of the GUI.

In the web layers (client and web server), we use Google Web Toolkit framework [39](GWT) to build a browser-based applications with Java programming language. GWT has a

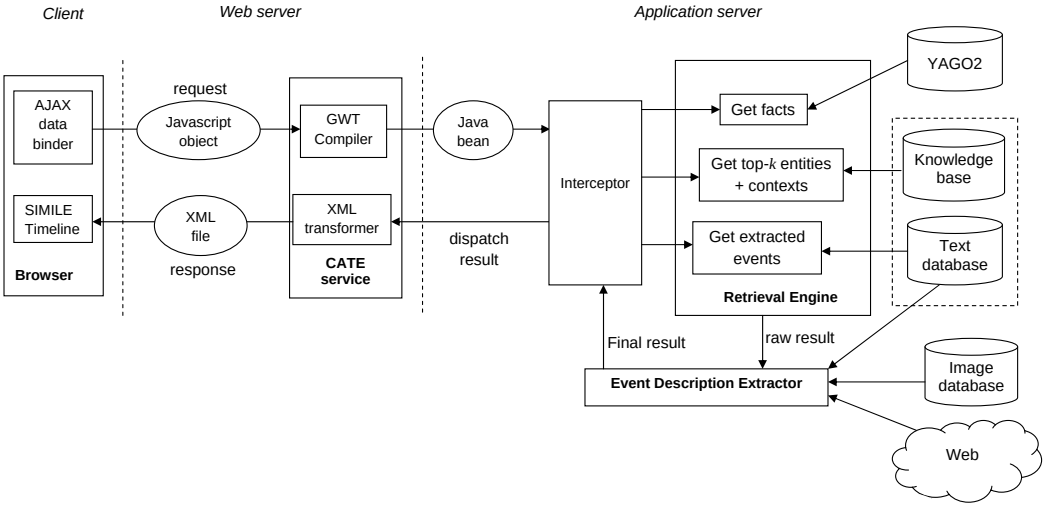


**Figure 7.1:** CATE's 3-tier Architecture

Java-to-Javascript compiler which can compile Java sources into pure Javascript and HTML in the client and vice versa, hence makes it easier for developers to develop and debug complex AJAX applications with Java technologies. We use SIMILE Timeline, a Javascript library for visualizing temporal data in browsers [40], to build our timeline illustrator as described in 4.1. Efforts have been made to adapt SIMILE Timeline to GWT framework seamlessly.

We use Apache Jetty server [41], a very light-weight HTTP server to deploy our GWT server components. These components communicate with the underlying application server components through Java remote objects (or Enterprise Java Bean - EJB). An interceptor is developed between two layers (Figure 7.1) to make all operations in the application server transparent to the Web server. The Web server sends a Java bean encapsulating all the input requests to the application server, and retrieves a Java bean encapsulating all the results. It then creates an XML file storing the results and sends it back to the timeline engine at the client side over the HTTP.

In the application layer, we implements two modules, the retrieve engine and the event-description extractor as described in the Section 4.2 and Section 4.3. The retrieval engine gets the facts of the entity of interest from YAGO, gets the events from our own populated database, and gets the top-9 contexts and entities within contexts from the knowledge base. The event-description extractor gets the ranked results from the retrieval engine, then connects to the text database and the image database to generate the events in a human-readable format (Section 7.4).

CATE works with two database servers: A Postgres server storing YAGO, and an Oracle server storing the text database and the image database. We decided to cache part of the YAGO knowledge base in our own database, so that the online module can connect to one server only.

## 7.2 Data Store

### 7.2.1 Text Database

The text database contains information about entities which is used in the retrieval engine. It extracted from the English Wikipedia dump, which is publicly available in two formats: a SQL dump and a XML dump [1]. The SQL dump stores meta-information about Wikipedia articles and categories (i.e., name, id, the latest modification, etc. ) in the table `page`; and stores information about hyperlinks between articles and about the grouping of articles into categories in two tables: `pagelinks` and `categorylinks` [2]. The XML dump is a single huge XML file (27 GB uncompressed) containing information about the full-text of all articles.

For the XML dump, we use Wikiprep [42], a tool written in Perl to preprocess the dump and to convert it into the formats that can be easily parsed by our extractors. Wikiprep splits the Wikipedia XML dump file into several medium-sized compressed files (around 128 MB each) called *chunks*. Each chunk contains a number of Wikipedia articles' full-text in a simple XML-like syntax, where unnecessary information (such as ids of users who changed each article, timestamps of article modifications) is removed, and other meta-data (such as link tables, category hierarchy, resolution of redirection links, etc.) are included [3].

For the SQL dump, we clean and store data in the two tables in the text database, *Entity Links* and *Entity Mapping*, as local indices (Figure 7.2). The table *EntityMapping* stores the meta-data of an entity in CATE: the corresponding Wikipedia article's id, location in the chunk where the full-text is stored, link frequency of the article (i.e., how many links point to the article from all other Wikipedia articles), link length (how many links appear in the article), an image URL (in case the image of the entity is not found in the image database). The *EntityLinks* stores hyperlinks between Wikipedia articles and the number of such links.

[1] The dump version we used in CATE is the English Wikipedia version on 30/07/2010, http://dumps.wikimedia.org/enwiki/20100730/

[2] The list of tables in Wikipedia SQL dump is available at http://www.mediawiki.org/wiki/Manual:Page_table

[3] http://www.cs.technion.ac.il/~gabr/resources/code/wikiprep/

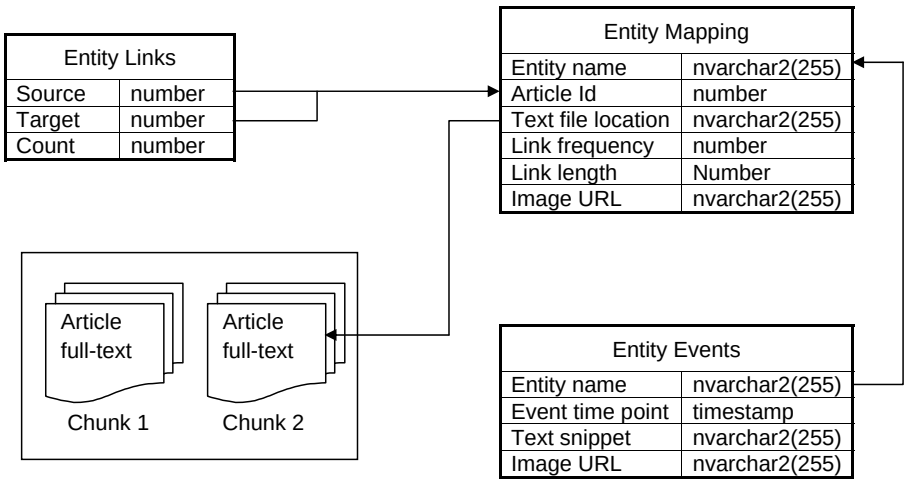


**Figure 7.2:** Excerpt of the text database schema

### 7.2.2 CATE Database

In CATE we store facts about entities and contexts in our own database, and cache partially facts in YAGO into one separate table. The schemas of the tables are depicted in Figure 7.3.

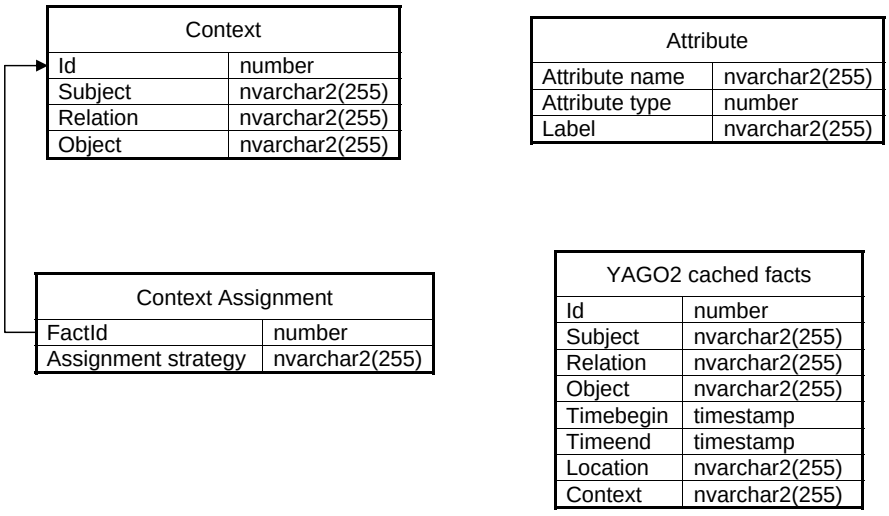


**Figure 7.3:** Excerpt of the CATE database schema

The table *Attribute* stores the names and the types of attributes extracted from the algorithms discussed in Section 5.1, as well as the labels to be displayed on the timeline. The names are mapped to YAGO entity names using our matching algorithms 5.1.3.

The table *Context* is a RDF table; it stores various facts about contexts mentioned in Section 3.3.1 in triples subject-relation-object. Some example facts for Gauss' case are shown in Table 7.1.

| id | Subject | Relation | Object |
|---|---|---|---|
| 124 | European_mathematicians | *hasLocation* | Europe |
| 125 | European_mathematicians | *hasTopic* | wordnet_mathematician_110301261 |
| 126 | 18-th_century_mathematicians | *hasTime* | "18##-##-##" |
| 127 | 18-th_century_mathematicians | *hasTopic* | wordnet_mathematician_110301261 |
| 128 | Differential_geometry | *hasTopic* | wikicategory_Differential_geometry |
| 129 | Carl_Friedrich_Gauss | *hasContext* | European_mathematicians |
| 130 | Carl_Friedrich_Gauss | *hasContext* | 18-th_century_mathematicians |
| 131 | Carl_Friedrich_Gauss | *hasContext* | Differential_geometry |

**Table 7.1:** Example facts in table *Context*

## 7.3 Implementation of Context Extraction

The facts in the table *Context* are populated using our context extraction algorithms in Section 8.3. The program runs periodically, and it runs in many iterations.

In the first iteration, we scan all the categories in Wikipedia dump, parse their names and map them to names in the table *Attribute*. This populates the table *Context* with with facts of type *hasTime*, *hasLocation* and *hasTopic*.

In the second iteration, we scan all entities in Wikipedia, extract their categories and check whether the categories are already included in the table *Context* from the first iteration. If found, we insert new facts of of type *hasContext* into the table *Context*.

In the third iteration, we extend the contexts based on their attributes. We group all the contexts of the same entity; and for each group, we combine the attributes of the contexts to create new contexts. For each location attribute, we extend to a more general attribute using *isLocatedIn* facts in YAGO and obtain new more contexts. Finally, we remove the duplicate contexts and add the new contexts to the table *Context*.

In the fourth iteration, we assign entities to contexts based on link strategy. We measure for each couple of a context $c$ and entity $e$ existing in the table *Context*, the confidence of assigning $e$ to $c$ using the equation 5.2. Our experiment (Chapter 8) suggests that setting $\alpha = 0.7$ in the equation 5.2 will give the most satisfactory context assignment. For each entity, we assign it to the top-5 contexts retrieved this way.

The above iterations require scanning the entire entities and categories in the Wikipedia many times, and are very time consuming. Hence, we implement our extraction program

using Map-Reduce paradigm to reduce the running time. We use Apache Hadoop framework [43, 44] to distribute the program on 32 cores. The total running time is within a day for the entire corpus.

## 7.4 Implementation of Event Description Extraction

As stated in Chapter 5, we have three main information extraction tasks, which are extracting facts about contexts, extracting texts for entities and events, and extracting images. To extract texts and images for the events of an entity, the system use two sources:

- the Wikipedia article of the entity
- the other articles which have links to the article of the entity

To extract events from the Wikipedia article of an entity, we develop a simple parser, which detect all sentences in the article. For each single sentence, we detect the timestamp expression and the presence of the entity name in the sentence. For example, given the article about Albert Einstein [3], the parser detects sentences such as, "In 1880, the family moved to Munich, where his father and his uncle founded Elektrotechnische Fabrik J. Einstein & Cie, a company that manufactured electrical equipment based on direct current". These sentences are then store the sentence with the timestamp in a table named *EntityEvent* (Figure 7.3). It contains the entity name associated with the event, the text snippet of the event and an image illustrating the event.

To get the image of an entity, we use using an image database extracted by the existing tool [28]. When no image for the entity is found, we use Google Image API [45] to issue query with the name of the entity to web search engine directly to get the URL of the top-1 image. To get the image for the event, we detect one entity in the text snippet of the event, and check the image of the entity in the image database. If such image is not found, we use Google Image API [45] to find one image URL on the Web and store it in the table *EntityEvent*.

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

## 7.5 Summary

In this chapter, we explain how we implement different components of CATE stated in Chapter 4. In the next chapter, we discuss some experiments that we conduct to evaluate the usefulness of these components.

# Chapter 8

# Experiments and Results

In order to evaluate the effectiveness of our proposed approach, we conducted some small-scale user studies. The experiments have been carried out to examine the information extraction component, as well as the ranking components mentioned in Chapter 6. The experiments were conducted on a computer with Intel Core 2 Duo 2.53 GHz and 3,45 GB working memory under Windows XP. The databases are stored in Oracle 11g and Postgres 8 database servers, and use Jetty HTTP server as an embedded Web server. Overall, we had 6 volunteers in our case study. The experimental results were sent to the volunteers in the form of text files to ask for their assessment. After collecting the feedback, the actual evaluation took place.

This chapter is organized as follows. Section 8.1 explains the setup. Section 8.2 discusses the evaluation measurement used. Section 8.3 describes the experiments and the results on context extraction part. Some experiments and results on the entity rankings are discussed in Section 8.4. In Section 8.5, we evaluated the event description extraction component. Finally, Section 8.6 demonstrates CATE in some scenarios.

## 8.1 Data Collection

We downloaded the Wikipedia SQL dump and XML dump and extracted a small excerpt from it. We collected 140 Wikipedia articles of sample entities, which are famous scientists (mathematicians, physicists, philosophers, architects, computer scientists, educationists, chemists, biologists, anthropologists), politicians, business people, artists, football players, actors and singers. These articles are *seeds* in our experiment. We then crawled all the Wikipedia articles that have incoming links to or outgoing links from the seeds. In addition, we collected all Wikipedia categories where the seeds belong to, and for

each category, we crawled all Wikipedia articles it contains. It gave us the considerable amount of articles and categories for our experimental purpose. Finally, we collected all the links between the Wikipedia articles. Our data set can be seen as a small sub-set of Wikipedia articles and categories network, centering around the seeds. Table 8.1 shows the size of our data set.

**Table 8.1:** Size of the data set

| Features | Size |
|---|---|
| Seeds set | 140 |
| Articles set | 881897 |
| Full categories set | 519592 |
| Leaf categories set | 403894 |
| Links between articles | 23051714 |

## 8.2 Evaluation Measurements

We introduce the measures used to evaluate our retrieval engine. There are several measures to assess the quality of an information retrieval, depending on a collection of documents and queries. We start with the most basic measures: precision and recall, then introduce more advanced measures such as Normalized Discounted Cumulative Gain (NDCG).

### 8.2.1 Precision and Recall

Precision and recall are two orthogonal metrics used for evaluating the goodness of an information retrieval algorithm. Precision measures the fidelity of a result (how exactly the system delivers the answer), while the recall measures the completeness of the result (how many correct answer the system covers) [1]. They are defined as follows [2]:

**Definition 8.1** (Precision)**.** Let $D$ be a set of documents, $R \subseteq D$ be the subset of relevant documents with respect to a query $q$, $A \subseteq D$ be the set of documents retrieved. The precision is the fraction of retrieved documents that are relevant:

$$Precision = \frac{|R \cap A|}{|A|} \tag{8.1}$$

**Definition 8.2** (Recall)**.** Let $D$ be a set of documents, $R \subseteq D$ be the subset of relevant documents with respect to a query $q$, $A \subseteq D$ be the set of documents retrieved. The recall

[1] http://en.wikipedia.org/wiki/Precision_and_recall
[2] These definitions are taken from [36, p. 142, 143 Chap. 8]

is the fraction of the documents that are relevant to the query and that are successfully retrieved:

$$Recall = \frac{|R \cap A|}{|R|} \tag{8.2}$$

The precision and recall are usually related by a trade-off: The high recall often leads to low precision and vice versa. In practice, an IR system can be characterized by the ratio of precision to recall, or a *precision-recall curve*.

### 8.2.2 Discounted Cumulative Gain (DCG)

Precision and recall are set-based measures. They do not distinguish the difference in order of documents retrieved. In reality, however, documents are delivered in the order of their relevance, and highly relevant documents are more valuable than marginally relevant documents. A recently adopted measure to overcome this issue is the *Discounted Cumulative Gain* [46] (DCG). DCG gives more weight to highly ranked documents, and it also allows incorporating different relevance levels, instead of binary relevance assessment (relevant or irrelevant) as in precision-recall measures.

**Definition 8.3** (Discounted Cumulative Gain)**.**

$$DCG(i) = \begin{cases} G(1) & \text{if } i = 1 \\ DCG(i-1) + \frac{G(i)}{\log(i)} & otherwise \end{cases} \tag{8.3}$$

where $i$ is the rank of the document retrieved, and $G(i)$ is the relevance level of the document.

In practice, DCG is often normalized to a value between 0 and 1. This is done by sorting the documents of a result list by relevance, producing an ideal DCG (where highly relevant documents are highly ranked as well). The normalized result is called a *Normalized Discounted Cumulative Gain*, or NDCG.

**Example 1.** Consider a query $q$ and 6 retrieved documents ordered by their ranks [1]. The relevance of the documents are in Table 8.2. Assuming the relevance level is defined as:

$$G(i) = \begin{cases} 3 & \text{if } d_i \text{ is highly relevant} \\ 2 & \text{if } d_i \text{ is relevant} \\ 1 & \text{if } d_i \text{ is slightly relevant} \\ 0 & \text{if } d_i \text{ is irrelevant} \end{cases}$$

[1] Example source: http://en.wikipedia.org/wiki/Discounted_cumulative_gain

The DCG for the whole result set will be:

$$DCG(6) = G(1) + \sum_{i=2}^{p} \frac{G(i)}{\log i} = 3 + (2 + 1.887 + 0 + 0.431 + 0.772) = 8.09$$

To normalize DCG values, an ideal ordering for the result of the query $q$ is needed; in this case, it is ($d_1$, $d_3$, $d_2$, $d_6$, $d_4$) (relevance level (3,3,2,2,1,0)). In this ideal result, the DCG is $8,693$, hence we have:

$$NDCG(6) = \frac{8.09}{8.693} = 0.9306$$

**Table 8.2:** Relevance judgment for results of the query $q$

| Documents | highly relevant | relevant | slightly relevant | irrelevant |
|---|---|---|---|---|
| $d_1$ | x | | | |
| $d_2$ | | x | | |
| $d_3$ | x | | | |
| $d_4$ | | | | x |
| $d_5$ | | | x | |
| $d_6$ | | x | | |

Teevan et al. [47] showed that using different gain values for highly relevant documents does not affect the NDCG computed (i.e., if $G(i)$ are defined differently, the overall result for NDCG will be the same).

### 8.2.3 Significance Tests

The natural way to compare the performance of two IR systems is to compare the means of assessment measures, such as the mean of NDCG over evaluated queries. However, if the data are small, the comparison result maybe misleading. To test if one system outperforms the other significantly, significance tests such as $t$-test are used instead.

A $t$-test is a statistical method used to test a null hypothesis that the means of two normally distributed populations are equal [3]. The test is done on two sample data sets (usually small) of the populations. The two data sets are characterized by data points, means and standard deviations, and we need to determine if the means are distinct or not. There are many variants of $t$-test. In this work, we used **paired t-test**, which tests on two samples of of the same or similar populations, where each point in one set has a unique relationship to one point in the other. Namely, we use paired $t$-test to test two results of the same set of queries using different algorithms.

[3] http://en.wikipedia.org/wiki/Student%27s_t-test

A paired $t$-test is done as follows. Given two paired populations $p_1$ and $p_2$, we want to decide whether the difference between their means are significant. We assume the difference is a random variable $t$ that follows the student's t-distribution. To compute the value of $t$, we collect two sample sets $X_1$ and $X_2$ of $p_1$ and $p_2$ respectively, and estimate $t$ on $X_1$ and $X_2$ as follows:

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\frac{S_d}{\sqrt{n}}} \tag{8.4}$$

with $n$-1 degrees of freedom, where $n$ is the size of two sample sets, $S_d$ is the standard deviation of the differences, and $\overline{X}_1$ and $\overline{X}_2$ are the averages of two sample sets. The value is calculated by equation 8.4 is called **$t$-value**. Now, we compute the probability that the variable $t$ is equal to the $t$-value given the certain degree of freedom, and call this probability **$p$-value**. If the $p$-value is below a threshold (significance level) in null hypothesis test (usually 0.05), we reject the hypothesis and claim that two sets are indeed distinct. In our contexts, it means that the retrieved result of one IR system really outperform the result of the other. The following example illustrates the computation of the paired $t$-test.

**Example 2.** Assume two sample sets $X_1$ and $X_2$ have 8 data points as shown in Table 8.3[1]. We have $\overline{X}_1 - \overline{X}_2 = 19.38$ and $S_d = 5.23$. Using equation 8.4, we have $t = 3.71$. The $p$-value is calculated using the standard $t$-distribution table with 6 degrees of freedom and is equal to 0.0076. Since $p$-value is less than the significance level (0.05), we reject the null hypothesis and conclude that two sets are actually significantly different; where the mean of the population represented by the set $X_1$ is greater than the mean of the population represented by the set $X_2$.

**Table 8.3:** Two sample sets for paired $t$-test

| id | $X_1$ | $X_2$ | $X1$ - $X_2$ |
|---|---|---|---|
| 1 | 162 | 168 | -6 |
| 2 | 170 | 136 | 34 |
| 3 | 184 | 147 | 37 |
| 4 | 164 | 159 | 5 |
| 5 | 172 | 143 | 29 |
| 6 | 176 | 161 | 15 |
| 7 | 159 | 143 | 16 |
| 8 | 170 | 145 | 25 |

[1]The data are taken from http://www.stattutorials.com

## 8.3 Experiments on Context Extraction

We conduct some experiments to evaluate the effectiveness of our context extraction algorithms. We evaluated our extraction algorithms in two criteria:

- How complete is the contexts extracted from categories ?
- How precisely does CATE construct the *hasContext* relationships based on different strategies ?

### 8.3.1 Category-based attribute and context extraction

This experiment aims to assess the capability of our rule-based and pattern learning-based algorithms in extracting contexts from the Wikipedia categories, as discussed in Section 5.1 and 8.3. For this purpose, we took Wikipedia categories as a ground truth and assumed the categories assigned to articles are correct, i.e. they imply all good contexts for an entity.

We sampled 28 entities and applied the used the category-based extraction algorithm to find contexts of these entities. Table 8.4 shows the example in the case of "Albert Einstein". Then we stored the results in separate text files; each consisted of the entity name together with the values of times, locations and topics extracted. For each file, we asked the participants to read the Wikipedia categories of the article of the entity of interest. Each participant then identified the potential values of time, location and topic, compared to the extracted values and counted the missing values.

**Result.** Table 8.5 shows the results for the the attribute extraction and context extraction based on Wikipedia categories. The second column indicates the proportion of Wikipedia categories extracted to build context sets. This set is based on participants' assessments as described above. Note that we only compare to the Wikipedia categories which directly group articles, i.e. leaf categories in the hierarchy. The third column is the precision, which was estimated by sampling the values of the attribute and evaluating manually. The precision of contexts extracted is not relevant here, since we assume Wikipedia categories as our ground truth, and we had one-to-one correspondence between categories and the contexts of this type.

Since the notion of context plays a key role in all parts of our work, we aimed to achieve a relatively low recall yet high precision of the attributes and context extraction procedures. Nevertheless, given the low recall of contexts extracted, the more advanced extraction algorithm is needed to improve the result in the future.

**Table 8.4:** Sample context extracted from categories of entity Albert_Einstein

| Categories extracted as contexts | Categories skipped |
|---|---|
| *German philosophers, Swiss philosophers, German scientists, Cosmologists, German inventors, American physicists, German physicists, Swiss physicists, American philosophers, American humanitarians, German humanitarians, Swiss humanitarians, People from Ulm, 19th-century German people* | *Academics of the Charles University, American pacifists, American Jews, American people of Swiss descent, American socialists, American vegetarians, Ashkenazi Jews, Deaths from abdominal aortic aneurysm, Deists, Einstein family, ETH Zurich alumni, ETH Zurich faculty, Fellows of the Leopoldina, Foreign Members of the Royal Society, German Jews who emigrated to the United States to escape Nazism, German Nobel laureates, Institute for Advanced Study faculty, Jewish agnostics, Jewish American scientists, Jewish American writers, Jewish inventors, Jewish pacifists, Jewish philosophers, Jewish refugees, Jewish scientists, Leiden University faculty, Members of the Prussian Academy of Sciences, Naturalized citizens of the United States, Nobel laureates in Physics, Patent examiners, People associated with the University of Zurich, People from the Kingdom of Württemberg, Recipients of the Gold Medal of the Royal Astronomical Society, Recipients of the Pour le Mérite (civil class), Religious skeptics, Stateless persons,...* |

**Table 8.5:** Evaluational result of time, location and topic extracted from Wikipedia categories

| | No. of values extracted | %categories extracted | precision |
|---|---|---|---|
| time | 713 | 19.4% | 100% |
| location | 209763 | 56.4% | 87.4% |
| topic | 696 | 15.1% | 95.8% |
| context | 29887 | 7.3% | |

### 8.3.2 Context Assignment

We investigated our algorithms of assigning entities to contexts based on the confidence-level score defined by equation 5.2. We also worked on 28 sample entities. First, we estimated the best parameter $\alpha$ for the confidence-level score. To do so, we set $\alpha = 0.0; 0.1; 0.2; 0.3; \ldots, 0.9; 1.0$ to obtain different values of the confidence-level score. For each version, we retrieved 28 sets of top-5 highest-scored contexts of 28 sample entities. To evaluate these sets, we changed the order of the result items arbitrarily and

added some random entities to make the results more noisy, then asked the participants to evaluate each result set manually. Each context was assessed against the entity of interest with two values: Relevant (1) or irrelevant (0).

**Table 8.6:** Sample contexts assigned to the entity Bernhard_Riemann

| Extracted from categories | Constructed by attribute expansion and combination | Assigned by link analysis |
|---|---|---|
| German_mathematicians<br>Differential_geometers<br>19th-century_mathematicians | European_mathematicians<br>19th-century_differential_geometers<br>19th-century_Europe<br>19th-century_Germany<br>19th-century_Number_theorists<br>19th-century_German_mathematicians | Mathematical_analysis<br>Differential_geometry<br>Fundamental_physics_concepts<br>Analytical_number_theory<br>Number_theorists |

**Estimation.** After collecting all the feedback, we estimated $\alpha$ as follows. For each value of $\alpha$, and for each context $c$ and entity $e$, we computed the average assessments of relevance of $c$ to $e$, and called this value $x(C, e)$. This value represents a random variable $X$. We estimated the correlation between this variable and the confidence-level score $conf(e, c)$ using *Pearson correlation*:

**Definition 8.4** (Pearson correlation)**.** Let $X$ and $Y$ be two random variables with expected values $e_x$ and $e_y$ and standard deviation $s_x$ and $s_y$ respectively. The Pearson correlation between $X$ and $Y$ is:

$$Pearson(X, Y) = \frac{E[(X - e_x)(Y - e_y)]}{s_x s_y}$$

where $E$ is the expected value operator.

Pearson correlation indicates the linear dependency between two random variables. We used this measure to estimate the goodness of different versions of the confidence-level score, in the sense that the best version will maximize the dependency between the rank of contexts (by confidence-level score) and the relevance of contexts to the entity (manually assessed). Table 8.7 shows the Pearson correlations of $X$ and the confidence-level scores at the different values of $\alpha$. From this table, we decided to set the parameter $\alpha = 0.7$.

**Evaluation.** After setting the parameter $\alpha$ for the confidence-level score, we evaluated the precision of the strategy of assigning entities to contexts based on this score. Based on the already assessed results of 28 sample entities, we obtained the precision of the strategy equal to **75.7%**.

**Table 8.7:** Result

| | $X = avg(relevance(c, e))$<br>$y = conf(e, c)$ |
|---|---|
| $\alpha = 0.0$ | $\text{Pearson}(x, y) = 0.3088$ |
| $\alpha = 0.1$ | $\text{Pearson}(x, y) = 0.3183$ |
| $\alpha = 0.2$ | $\text{Pearson}(x, y) = 0.3296$ |
| $\alpha = 0.3$ | $\text{Pearson}(x, y) = 0.3432$ |
| $\alpha = 0.4$ | $\text{Pearson}(x, y) = 0.3594$ |
| $\alpha = 0.5$ | $\text{Pearson}(x, y) = 0.3779$ |
| $\alpha = 0.6$ | $\text{Pearson}(x, y) = 0.3967$ |
| $\alpha = \mathbf{0.7}$ | $\mathbf{Pearson(x, y) = 0.4078}$ |
| $\alpha = 0.8$ | $\text{Pearson}(x, y) = 0.3923$ |
| $\alpha = 0.9$ | $\text{Pearson}(x, y) = 0.329$ |
| $\alpha = 1.0$ | $\text{Pearson}(x, y) = 0.231$ |

## 8.4 Experiments on Entity Retrieval

**Experiment setting.** We conducted a number of experiments to examine our module for ranking entities relevant to a given entity within a certain context. In each experiment, we collected 28 entities and extracted two contexts for each. Each individual entity and context was then used as a query to retrieve the top-5 most relevant entities to the entity within the context. To evaluate the results, we re-arranged the items in each result set in random order to avoid the position bias. We also inserted some random entities to make the results more noisy, and sent them to the participants in the form of text files. Each text file stores a list of relevant and randomly inserted entities, and the name of the entity of interest and the context. Each participant chose some entities about which they had certain knowledge. The participant were asked to mark each item in the results with three different values: Highly relevant, relevant and completely irrelevant.

**Experiment 1.** We investigated the effect of adopting ranking paradigms on the result. We issued queries with the name of one of the 28 sample entities and one corresponding context, and configured CATE to return results in two different orders. The first order version returns the results as it is stored in the database, i.e. random order or no ranking. The second order is based on the ranks of entities, adopting ranking function described by equation 6.12, with Dirichlet smoothing parameter $\lambda = 0.5$. Table 8.8 shows example of results in the second order.

**Result.** We collected the files from all participants. For each query, we computed the NDCG (Section 8.2.2) of the assessed results. We used $G(i) = 2$ for highly relevant

**Table 8.8:** Example results for the query "Michael Jackson, in context: American dance musicians"

| Top-5 results | Relevance score |
|---|---|
| Madonna_(entertainer) | 0.1767 |
| Janet_Jackson | 0.1745 |
| Whitney_Houston | 0.1194 |
| Quincy_Jones | 0.1032 |
| Stevie_Wonder | 0.1023 |

entities, $G(i) = 1$ for relevant entities and $G(i) = 0$ for irrelevant entities. We compared the average NDCG of different ranking paradigms.

As shown in Figure 8.1, statistical-language-model ranking significantly improves the quality of results. The average NDCG value of ranking-based results was significantly higher than the average NDCG value of random-based results. The experiment supports our belief that adopting ranking really enhances the quality of the retrieval engine and better satisfies users' information needs.

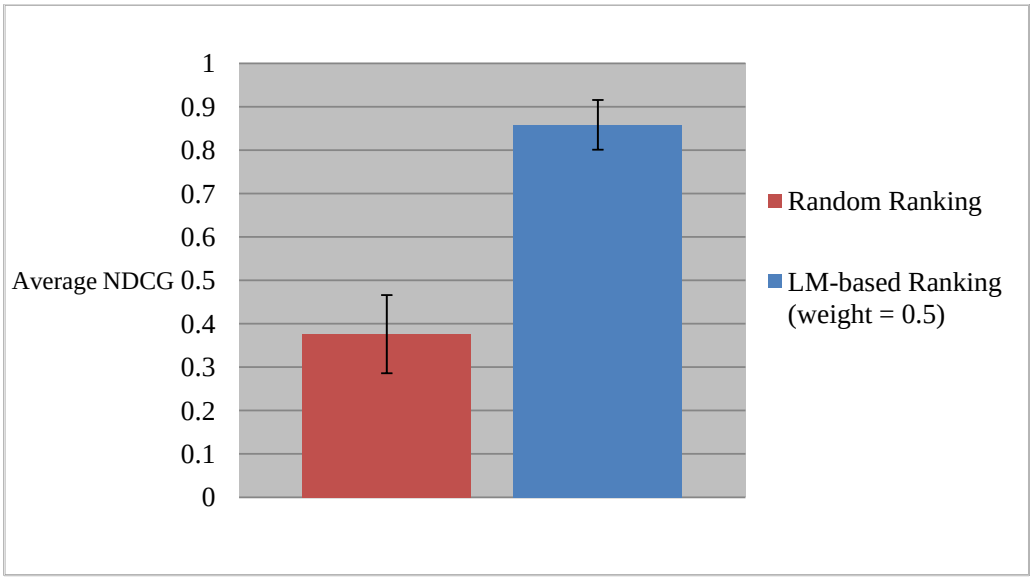


**Figure 8.1:** Average NDCG for statistical-language-model ranking versus random ranking

Since the size of the sample data is small, we conducted a paired $t$-test to test whether the improvement by the ranking was significant as our intuition. Table 8.9 shows the results of the $t$-test. The $t$-value is equal to $-7.45401$ and the $p$-value (two tail) is equal to 3.88E-05, which is much less than the significance level (0.5). Then we rejected the null hypothesis and concluded that the mean of the NDCG value of the ranking-based results is indeed higher than the mean of that of the random-based results.

**Table 8.9:** Paired $t$-test for comparison statistical-language-model ranking versus random ranking

| | Variable 1 | Variable 2 |
|---|---|---|
| Mean | 0.376088 | 0.8584 |
| Variance | 0.030211 | 0.003283 |
| Observations | 8 | 8 |
| Hypothesized Mean Difference | 0 | |
| Degrees of Freedom | 9 | |
| t-value | -7.45401 | |
| $P(T \leq t)$ one-tail | 1.94E-05 | |
| t Critical one-tail | 1.833113 | |
| $P(T \leq t)$ two-tail | 3.88E-05 | |
| t Critical two-tail | 2.262157 | |

**Experiment 2.** In this experiment, we studied the impact of smoothing in our ranking model. We used the same setting, but each query (composed of an entity and a context) has a different result set with different values of the smoothing parameter. As shown in Figure 8.2, the average NDCG did not change much when the smoothing parameter changed from 0.1 to 0.9, but at the point 0.0 (no smoothing), the average NDCG reduced considerably.

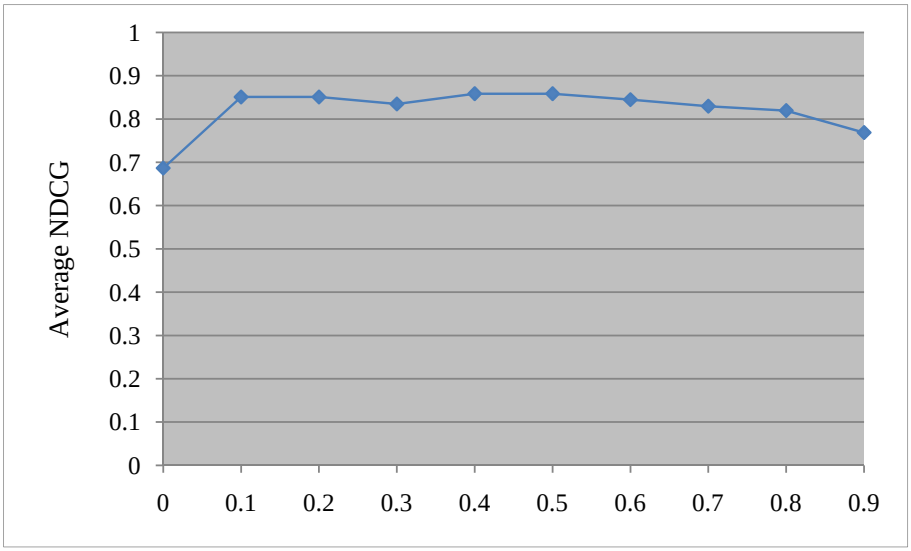


**Figure 8.2:** Changes of average NDCG for the statistical-language-model rankings with different smoothing weights

Due to the scale of the experiment, we also conducted a paired $t$-test to confirm our assumption. We compare the ranking without smoothing and the ranking with smoothing weight $\lambda = 0.5$. Table 8.10 shows that $t$-value is equal to $-4.90532$ and $p$-value is equal

to 0.000287. This value is much less than the significance level 0.05, which means that our argument of using smoothing to avoid overfitting problem indeed worked.

**Table 8.10:** Paired $t$-test for comparison statistical-language-model ranking without smoothing versus with smoothing $\lambda = 0.5$

| | Variable 1 | Variable 2 |
|---|---|---|
| Mean | 0.686625 | 0.8584 |
| Variance | 0.006527 | 0.0032383 |
| Observations | 8 | 8 |
| Hypothesized Mean Difference | 0 | |
| Degrees of Freedom | 13 | |
| t-value | -4.90532 | |
| $P(T \leq t)$ one-tail | 0.000144 | |
| t Critical one-tail | 1.770933 | |
| $P(T \leq t)$ two-tail | 0.000287 | |
| t Critical two-tail | 2.160369 | |

**Experiment 3.** In this experiment, we studied how our context-awareness affect the ranking model in general. For this purpose, we developed a "non-context-aware" version of our ranking model. Given $e$ which is an entity of interest, we want to rank the entity $e'$ based on the probability of generating the entity $e$ given the article of the entity $e'$, which is calculated using equation 6.8 (we set $\lambda = 0.5$). Table 8.11 shows the result of the query "Bill Gates, in context American businesspeople" as compared to the result of the query "Bill Gates".

**Table 8.11:** Results of query Bill_Gates with and without context

| Query 1: Bill_Gates | | Query 2: Bill_Gates, context: American_businesspeople | |
|---|---|---|---|
| Top-5 results | relevance | Top-5 results | relevance |
| Microsoft | 0.0172 | Steve_Job | 0.1124 |
| International_Standard_Book_Number | 0.0125 | John_D._Rockefeller | 0.0209 |
| Bill_&_Melinda_Gates_Foundation | 0.0107 | Andrew_Carnegie | 0.0217 |
| BBC_News | 0.0066 | Paul_Allen | 0.0208 |
| Paul_Allen | 0.0064 | David_Rockefeller | 0.0189 |

We also conducted experiments with 28 sample entities. For each entity of interest, the participants were presented with the result of relevant entities to the entity as computed by equation 6.8. The participant then chose one context to see the new result of the relevant entities to the given entity within the chosen context. This time, the participant assessed each item in both results with only two values: Possibly relevant (1) or completely irrelevant (0).

Figure 8.3 shows the comparison in average precision of results returned by context-aware and non context-aware ranking procedures. Although it is difficult to claim that the introduction of context in ranking significantly improves the quality of the results, since the two ranking paradigms aim to fulfill different types of user information need, we can at least see that the presence of contexts in a query enables the user to express their information need more precisely, thereby obtaining results with higher precision (eliminate more irrelevant results).

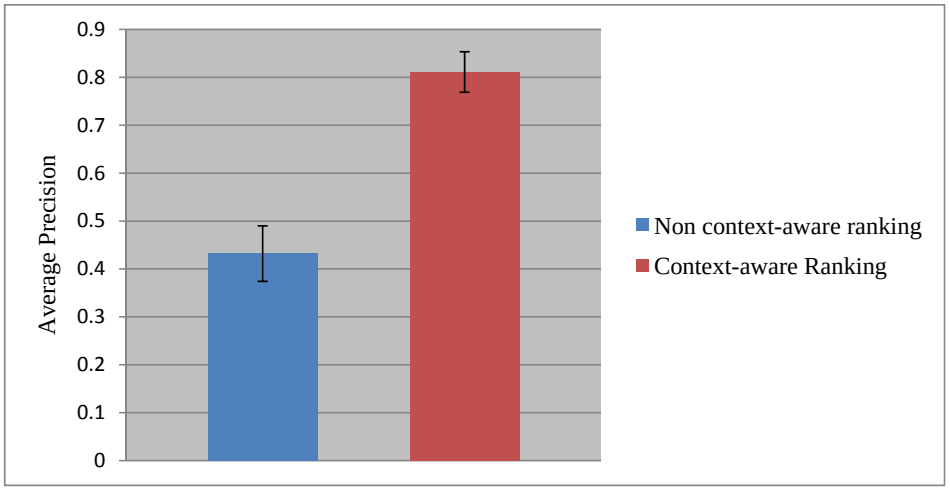


**Figure 8.3:** Comparison between precision of non context-aware ranking and context-aware ranking

## 8.5 Experiments on Event Description Extraction

The direct events for an entity are extracted from three sources: Facts about the entity in the YAGO knowledge base, text in the Wikipedia article of the entity, and text snippets of other Wikipedia articles linking to the entity of interest. Table 8.12 shows the example events of Albert Einstein as extracted from these sources. In this experiment, we investigated how well our event description extractor works in two criteria:

- whether the sentence directly states one event during the life of the entity and,
- whether the time point correctly capture the time when the event occur ?

We conducted experiments to evaluate the precision of our different extraction strategies. We also worked on 28 sample entities. For each entity, we extracted events using three

**Table 8.12:** Example events of Albert Einstein as derived from different sources

| **Events extracted from YAGO facts** |
|---|
| **14-03-1879**: Einstein was born in Ulm<br>**1903**: Einstein was married to Mileva Marić<br>**14-05-1904**: Einstein had a child Hans Albert Einstein<br>**28-07-1910**: Einstein had a child Eduard Einstein<br>**1921**: Einstein won Nobel Prize in Physics<br>... |
| **Events extracted from Albert Einstein's Wikipedia article** |
| **1880**: In 1880, the family moved to Munich, where his father and his uncle founded "Elektrotechnische Fabrik J".<br>**1894**: In 1894, his father's company failed: direct current (DC) lost the War of Currents to alternating current (AC).<br>**1896**: At age 17, he graduated, and, with his father's approval, renounced his German citizenship in the German Kingdom of Württtemberg to avoid Conscription in Germany military service, and in 1896 he enrolled in the four year mathematics and physics teaching diploma program at the Polytechnic in Zurich.<br>**30-04-1905**: On 30 April 1905, he completed his thesis, with Alfred Kleiner, Professor of Experimental Physics, serving as pro-forma advisor.<br>... |
| **Events extracted from other Wikipedia articles** |
| **27-09-1905**: The physics journal "Annalen der Physik" published Albert Einstein's paper "Does the Inertia of a Body Depend Upon Its Energy Content?", introducing the equation $E = mc^2$.<br>**29-05-1919**: The confirmation is announced of Einstein's general relativity theory, tested by Arthur Eddington and Andrew Crommelin during a total solar eclipse on May 29, 1919 FirstScience.<br>**17-10-1933**: Albert Einstein arrives in the United States as a refugee from Nazi Germany<br>**24-01-1934**: Einstein visits the White House.<br>**11-10-1939**: Manhattan Project: U.S. President Franklin D. Roosevelt is presented a letter signed by Albert Einstein, urging the United Statesto rapidly develop the atomic bomb.<br>... |

sources, then generated sentences in natural languages (each sentence is marked with a time point) and stored all of them in a single text file. The participants were asked to read the sentence and evaluate two things:

**Result.** Figure 8.4 shows the average precision of the events extracted in three ways. As we can see, the precision of results extracted from facts in YAGO is highest as compared to the other two. This is because of our filtering heuristics: we only retrieved from YAGO facts with certain relation names and defined rules to group similar facts into one event. The precision of the events extracted from other Wikipedia articles is also very high, due to use of links between articles to resolve the named entity disambiguation in sentences. The precision of the events extracted from the entity's article is the lowest, due to the simplified implementation of the text parser (Section 7.4), even though the number of events extracted this way is much higher (Table 8.13). We plan to enhance this parser in the next version of CATE.

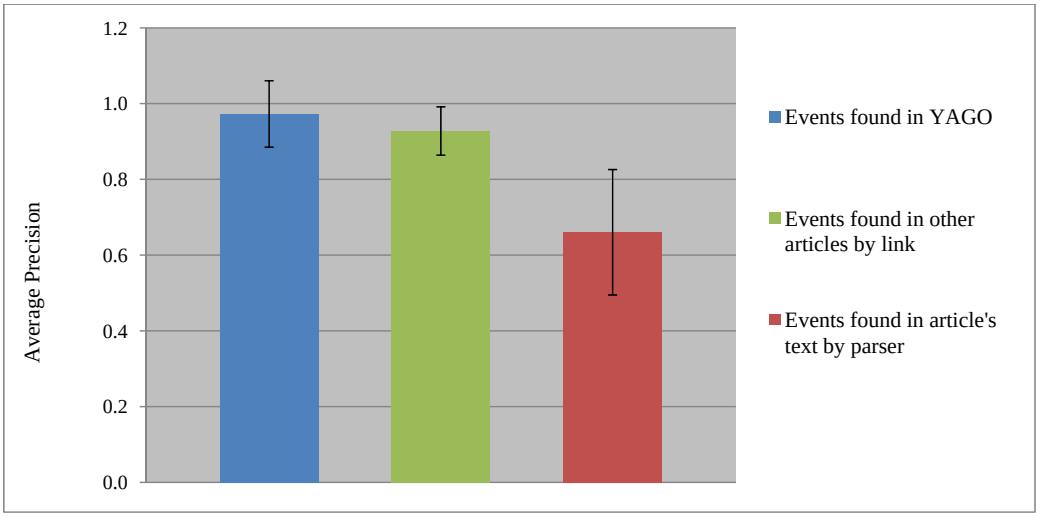


**Figure 8.4:** Comparison between precision of non context-aware ranking and context-aware ranking

One interesting observation is that the distribution of events varies depending on the domain of the entity. As shown in Table 8.13, entities about people who have significant influence such as Napoleon I have many events mentioned in other Wikipedia articles. On the other hand, entities about modern famous people or celebrities (e.g, Elvis Presley or Frank Sinatra) have much more events presented in structured format (as extracted by YAGO) rather than presented in free text (as extracted by the parser). This implies that the recent Wikipedia articles about famous people are more well-structured (probably

because of the bigger community of contributors), while the Wikipedia articles about historic people have a higher degree of incoming links (when the Wikipedia articles evolves).

**Table 8.13:** Distribution of events from sources of example entities

| Entity | Events from YAGO | Events No. from entity's article | Events from other articles |
|---|---|---|---|
| Albert_Einstein | 9 | 66 | 25 |
| Nelson_Mandela | 6 | 58 | 12 |
| Bill_Gates | 4 | 32 | 6 |
| Elvis_Presley | 180 | 42 | 22 |
| Frank_Sinatra | 173 | 74 | 2 |
| Napoleon_I | 5 | 70 | 108 |

## 8.6 Demonstration Scenarios

We demonstrate the main work-flow of CATE. As stated in Chapter 4, the GUI uses a text box to take the name of an entity as an input. In the first scenario, the user types the name "Albert Einstein" into the textbox. CATE connects to the retrieval engine and gets the result in form of an XML file. This file is rendered on the timeline illustrator, as shown in Figure 8.5. Each item in the timeline represents an event and has three elements: A text snippet, an image and a time point. When an appropriate image of the event is not found, a small icon is replaced. The user can use the two bands (indicated by the red arrows in Figure 8.5) to move the timeline to check other events.

In the second scenario, the user chooses the attribute values to interpret her inquiry of contexts of Albert Einstein. The system then tries to find the context that best matches the attributes. If no context is found, an error message is displayed in the right-handed side on the text box . Otherwise, the system retrieves the top-5 most relevant entities to Albert Einstein and the chosen context. These entities are displayed together with other events of Albert Einstein in the timeline illustrator. Figure 8.6 shows the snapshot of the GUI when the user chooses to explore Albert Einstein's life within the context "Germany, physicists". Besides his events, the images and short descriptions of other related German physicists (Max Planck, Niels Bohr, Arnold Sommerfeld, Werner Heisenberg, Max Born) are shown.

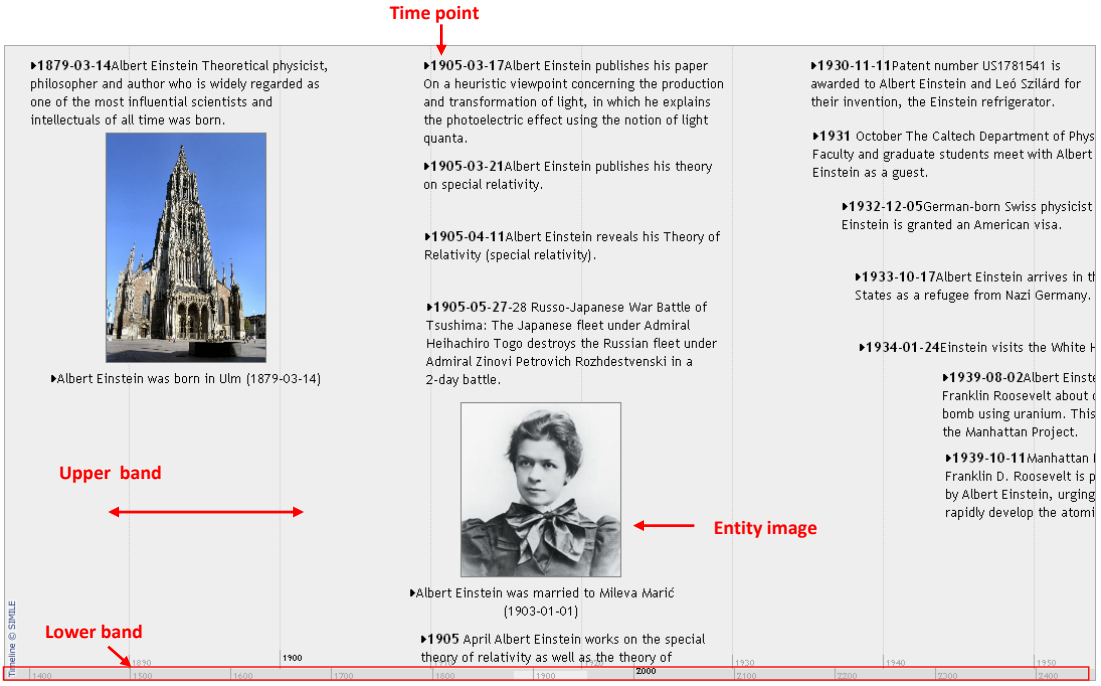


**Figure 8.5:** Snapshot of the Timeline Illustrator when the user explores "Albert Einstein"

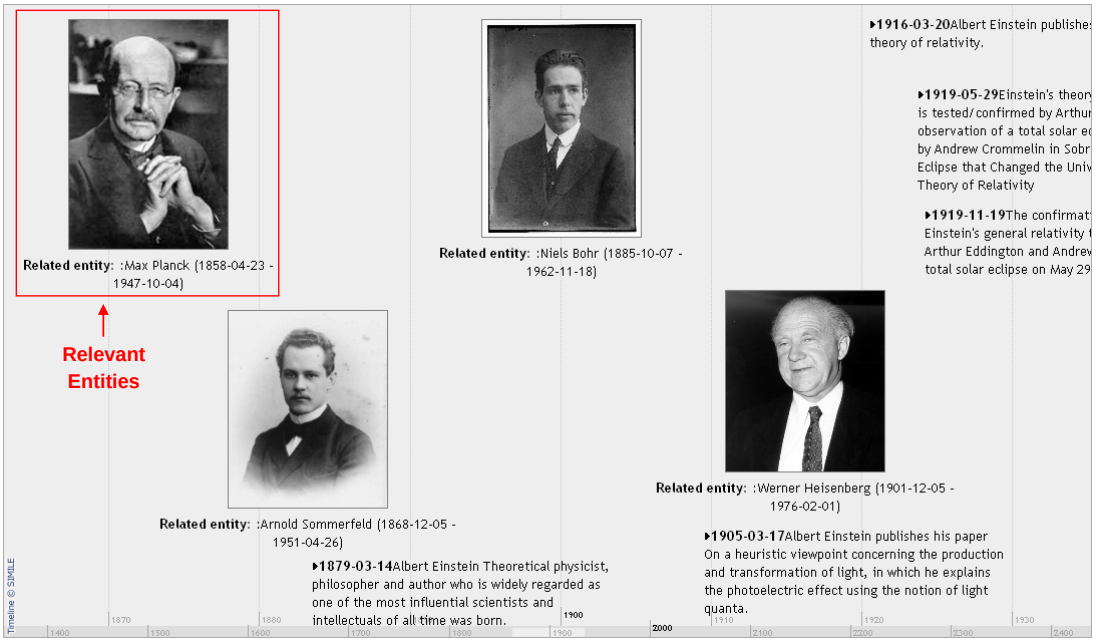


**Figure 8.6:** Snapshot of the Timeline Illustrator when the user explores "Albert Einstein" within the context "Germany, physicists"

# Chapter 9

# Conclusion and Future Work

## 9.1 Conclusion

In this thesis, we have presented a framework to extract events of an entity in Wikipedia and to visualize them in a timeline fashion. We have introduced the notion of context to comprehensively capture many aspects of an entity, hence creating an informative portrait of the entity. We have extended the YAGO data model to represent context information in a consistent manner. We have developed several algorithms to extract new facts from Wikipedia which can be integrated into the YAGO at minimum effort. We have also implemented an end-to-end system to visualize an entity's life in an intuitive and appealing manner.

Additionally, we have proposed a novel ranking model to retrieve relevant entities to a given entity. Our ranking model is context-aware and based on the current-state-of-the-art statistical-language-modeling. Preliminary experiments show the effectiveness of our ranking model in retrieving the most relevant entities to satisfying user various interests.

## 9.2 Future Work

Our framework has been developed using baseline solutions and has rooms for improvements in the future. We see some possible directions of future work as follows:

- **Constructing contexts as a full-fledged ontology.** In our current work, contexts are defined by attributes, and attributes are considered independent. By developing a complete relationship network between attributes and contexts, we

can develop a more sophisticated paradigm to explore entities' lives through many facades.

- **Predicting categories for an article.** We have defined a strategy to assign articles to contexts based on link analysis. At the moment, we only consider the links between articles and the relationships between articles and categories. We hope to devise more advanced techniques, utilizing other features of Wikipedia (redirects, disambiguation pages, etc.) to improve the quality of the assignment. The new strategy could provide a powerful way to predict potential categories for a certain Wikipedia article, thereby supporting further mining tasks.

- **Improving context extraction procedures.** Our baseline solution construct contexts using Wikipedia categories. As shown in Section 8.3, the context set only covers a small proportion of Wikipedia categories, which means there are still a large number of potential categories to be mined. Furthermore, we have an ambition to construct contexts using many other sources in Wikipedia (free-text, info-boxes, anchor texts, etc.).

- **Improving entity ranking.** Our current ranking model uses article link networks as a primary source. However, we believe this is insufficient to discover the relationships between entities. We see at least three ways to extend our ranking model: 1) Exploiting article text and combing with traditional ranking techniques in IR; 2) Extracting the semantic relatedness between entities and label links to classify the link graph; 3) Putting new features in ranking, for instance the explicit relationships between entities stored in the knowledge base.

- **Improving event extraction.** We currently connect to the YAGO knowledge base, and develop simplified strategies to extract events from the Wikipedia article of an entity as well as from other Wikipedia articles over the entire corpus. In the future, we plan to develop more sophisticated tools (deep parser, Wikipedia link graph analyzer) to extract richer set of events with higher precision. We also plan to tap other RDF-style knowledge bases in the linked data cloud to expand the results significantly

- **Exploring images with time information** One interesting extension in future work is the integration of time information with our image extraction. For example, to display an image of an entity (e.g., Trocadero) in a specific point in time (e.g., "1900"), we can issue a query to web search engine with additional time information (for example, "Trocadero 1900") to get a more exact image of the entity at the given time. As illustrated in Figure 9.1, two queries of Tocadero with and without time information can have siginificantly different results. We can even go further

by integrating the image search of a time travel search to retrieve the entity image within certain timepoints.

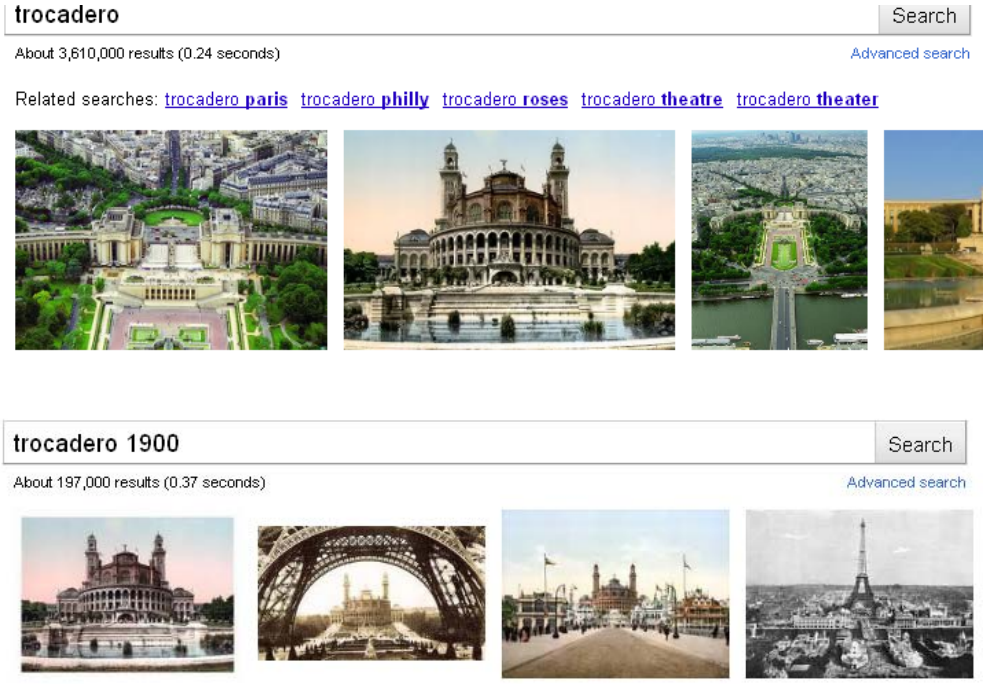


**Figure 9.1:** Image results of the entity Trocadero with and without the timepoint (1900)

- **Integrating CATE with a bigger knowledge harvesting framework.** CATE has a capability to retrieve events from additional sources other than info-boxes, and it provides a ranking model to define implicit relationships between entities. We see this valuable for the integration of CATE with a bigger knowledge harvesting framework, where CATE's extraction results can be used by other IE component to acquire new facts; or can be used as a debugging framework to examine internal IE tasks. At even more universal scenario, CATE can be re-engineered to be a generic framework for knowledge exploration, which any knowledge-based system can use to discover its dynamics.

# List of Figures

# List of Tables

# Bibliography


[1] F. Song and W. B. Croft. A general language model for information retrieval. In *SIGIR*, 1999.

[2] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.

[3] Nicoleta Preda, Fabian M. Suchanek, Gjergji Kasneci, Thomas Neumann, Maya Ramanath, and Gerhard Weikum. Angie: active knowledge for interactive exploration. *Proc. VLDB Endow.*, 2, August 2009.

[4] Michiel Hildebrand, Jacco van Ossenbruggen, and Lynda Hardman. /facet: A Browser for Heterogeneous Semantic Web Repositories. In *ISWC*, 2006.

[5] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, Berlin, Heidelberg, 2007. Springer-Verlag.

[6] Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 635–644, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2.

[7] Christiane Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.

[8] Vivi Nastase and Michael Strube. Decoding wikipedia categories for knowledge acquisition. In *AAAI*, 2008.

[9] Peter Schonhofen. Identifying document topics using the wikipedia category network. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '06, Washington, DC, USA, 2006. IEEE Computer Society.

[10] Qi Zhang, Fabian M. Suchanek, Lihua Yue, and Gerhard Weikum. TOB: Timely Ontologies for Business Relations. In *11th International Workshop on Web and Databases 2008 (WebDB 2008)*. ACM, 2008.

[11] J. Pustejovsky, J. Castano, R. Ingria, R. Sauri, R. Gauzauskas, A. Setzer, and G. Katz. Timeml: Robust specification of event and temporal expression in text. *IWCS-5, Fifth International Workshop on Computational Semantics.*, 2003.

[12] Yafang Wang, Mingjie Zhu, Lizhen Qu, Marc Spaniol, and Gerhard Weikum. Timely yago: harvesting, querying, and visualizing temporal knowledge from wikipedia. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 697–700, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-945-9.

[13] Hugo Zaragoza, Henning Rode, Peter Mika, Jordi Atserias, Massimiliano Ciaramita, and Giuseppe Attardi. Ranking very many typed entities on wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 1015–1018, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9.

[14] Guus Schreiber, Alia Amin, Mark van Assem, Victor de Boer, Lynda Hardman, Michiel Hildebrand, Laura Hollink, Zhisheng Huang, Janneke van Kersen, Marco de Niet, Borys Omelayenko, Jacco van Ossenbruggen, Ronny Siebes, Jos Taekema, Jan Wielemaker, and Bob Wielinga. Multimedian e-culture demonstrator. In *In the Semantic Web Challenge at the 5th International Semantic Web Conference*, 2006.

[15] Gianluca Demartini, Tereza Iofciu, and Arjen P. de Vries. Overview of the inex 2009 entity ranking track. In *INEX*, pages 254–264, 2009.

[16] Anne-Marie Vercoustre, James A. Thom, and Jovan Pehcevski. Entity ranking in wikipedia. In *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, pages 1101–1106, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-753-7.

[17] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. Language-model-based ranking for queries on rdf-graphs. In Wesley Chu, Xiaohua Hu, and Jimmy Lin, editors, *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, page ., Hongkong, China, 2009. ACM.

[18] Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, and Gerhard Weikum. Naga: Searching and ranking knowledge. *Data Engineering, International Conference on*, 0, 2008.

[19] Zaiqing Nie, Yunxiao Ma, Shuming Shi, Ji-Rong Wen, and Wei-Ying Ma. Web object retrieval. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, New York, NY, USA, 2007. ACM.

[20] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. World Wide Web Consortium, Recommendation, February 2004. URL http://www.w3.org/TR/2004/REC-rdf-concepts-20040210.

[21] Fabian M. Suchanek. *Automated Construction and Growth of a Large Ontology*. PhD thesis, Saarland University, 2009.

[22] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (uri): Generic syntax, 1998.

[23] RDFS. RDF Vocabulary Description Language 1.0: RDF Schema, February 2004.

[24] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A Large Ontology from Wikipedia and WordNet. *Elsevier Journal of Web Semantics*, 2008.

[25] Paul V. Biron and Ashok Malhotra. XML Schema Part 2: Datatypes. Recommendation, World Wide Web Consortium, May 2001. See http://www.w3.org/TR/xmlschema-2/.

[26] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. Research Report MPI-I-2010-5-007, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, November 2010.

[27] Shady Elbassuoni, Katja Hose, Steffen Metzger, and Ralf Schenkel. Roxxi: Reviving witness documents to explore extracted information. *Proc. VLDB Endow.*, 3, September 2010.

[28] Bilyana Taneva, Mouna Kacimi, and Gerhard Weikum. Gathering and ranking photos of named entities with high precision, high recall, and diversity. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, New York, NY, USA, 2010. ACM.

[29] Evgeniy Gabrilovich and Shaul Markovitch. Overcoming the brittleness bottleneck using wikipedia: enhancing text categorization with encyclopedic knowledge. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, pages 1301–1306. AAAI Press, 2006. ISBN 978-1-57735-281-5.

[30] ISO. *ISO 8601:1988. Data elements and interchange formats — Information interchange — Representation of dates and times*. 1988. URL http://www.iso.ch/cate/d26780.html.

[31] Dan Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA, 1997. ISBN 0-521-58519-8.

[32] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf (working draft). Technical report, W3C, March 2007. URL http://www.w3.org/TR/2007/WD-rdf-sparql-query-20070326/.

[33] Fabian M. Suchanek, Georgiana Ifrim, and Gerhard Weikum. Leila: Learning to extract information by linguistic analysis. In *Second Workshop on Ontology Population (OLP2) at ACL/COLING*, 2006.

[34] Sergey Chernov, Tereza Iofciu, Wolfgang Nejdl, and Xuan Zhou. Extracting semantics relationships between wikipedia categories. In Max Völkel and Sebastian Schaffert, editors, *SemWiki*, volume 206 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[35] Djoerd Hiemstra. Language models. In *Encyclopedia of Database Systems*, pages 1591–1594. 2009.

[36] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

[37] Larry Wasserman. *All of statistics*. Springer, Berlin, Heidelberg, 2004. ISBN 0387402721, 9780387402727.

[38] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22:179–214, April 2004. ISSN 1046-8188.

[39] Google Inc. Google web toolkit framework. Google Code Directory, 2011. URL http://code.google.com/webtoolkit/.

[40] David Huynh. Simile timeline, web widget for visualizing temporal data. SIMILE Widgets project, 2011. URL http://www.simile-widgets.org/timeline/.

[41] David Huynh. Apache jetty framework, 2009. URL http://jetty.codehaus.org/jetty/.

[42] Evgeniy Gabrilovich Tomaz Solc, Chris Jordan. Wikipedia preprocessor, 2010. URL http://sourceforge.net/projects/wikiprep/.

[43] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, 2009. ISBN 0596521979, 9780596521974.

[44] Apache Software Foundation. Apache hadoop, 2011. URL http://hadoop.apache.org/.

[45] Google Inc. The images java api, 2009. URL http://code.google.com/appengine/docs/java/images/.

[46] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, New York, NY, USA, 2000. ACM.

[47] Jaime Teevan, Susan T. Dumais, and Eric Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, New York, NY, USA, 2005. ACM.