Semantic Analysis of Wikipedia's Linked Data Graph for Entity Detection and Topic Identification Applications

by

Milad AlemZadeh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2012

AUTHOR'S DECLARATION

Abstract

Semantic Web and Linked Data community is now the reality of the future of the Web. The standards and technologies defined in this field have opened a strong pathway towards a new era of knowledge management and representation for the computing world. The data structures and the semantic formats introduced by the Semantic Web standards offer a platform for all the data and knowledge providers in the world to present their information in a free, publicly available, semantically tagged, inter-linked, and machine-readable structure. As a result, the adaptation of the Semantic Web standards by data providers creates numerous opportunities for development of new applications which were not possible or, at best, hardly achievable using the current state of Web which is mostly consisted of unstructured or semi-structured data with minimal semantic metadata attached tailored mainly for human-readability.

This dissertation tries to introduce a framework for effective analysis of the Semantic Web data towards the development of solutions for a series of related applications. In order to achieve such framework, Wikipedia is chosen as the main knowledge resource largely due to the fact that it is the main and central dataset in Linked Data community. In this work, Wikipedia and its Semantic Web version DBpedia are used to create a semantic graph which constitutes the knowledgebase and the back-end foundation of the framework. The semantic graph introduced in this research consists of two main concepts: entities and topics. The entities act as the knowledge items while topics create the class hierarchy of the knowledge items. Therefore, by assigning entities to various topics, the semantic graph presents all the knowledge items in a categorized hierarchy ready for further processing.

Furthermore, this dissertation introduces various analysis algorithms over entity and topic graphs which can be used in a variety of applications, especially in natural language understanding and knowledge management fields. After explaining the details of the analysis algorithms, a number of possible applications are presented and potential solutions to these applications are provided. The main themes of these applications are entity detection, topic identification, and context acquisition. To demonstrate the efficiency of the framework algorithms, some of the applications are developed and comprehensively studied by providing detailed experimental results which are compared with appropriate benchmarks. These results show how the framework can be used in different configurations and how different parameters affect the performance of the algorithms.

Acknowledgements

This work has only been made possible with the unwavering support and help of my supervisor Professor Fakhreddine Karray to whom I would like to give my sincere thanks. I am forever grateful to him for the freedom he provided me to explore the areas of science I have always liked to venture into while assisting me whenever I needed it. For this and all the directions I received throughout all the stages of this work, I am thankful to him.

I would also like to express my gratitude to the members of my committee, Professors Robert Mercer, Mahesh Tripunitara, Kshirasagar Naik, and Kumaraswamy Ponnambalam for all of their invaluable comments and suggestions which helped improve the quality of this work.

I would like to extend my appreciation to Professors Frank Tompa and Mohamed Kamel for the valuable lessons I learned in their classes which helped shape the work presented in this dissertation.

Moreover, I would very much like to acknowledge the vital role my family has played in helping me achieve my goals in life. I am particularly thankful to my wife, Sarah, for her relentless support these past few years and to my mother for providing a great childhood in spite of all the hardships. I am very grateful for the unconditional support of my sisters, Parisa and Soraia, and my brothers, Davood and Saeed, whenever I needed it.

Last but certainly not least, I would like to thank my good friend and the co-author of many of my published works Professor Richard Khoury. In collaboration with Richard, I learned how to better express my ideas and, at the same time, improve my work considerably. This work would have undoubtedly not been possible if it was not for his tireless efforts and significant feedback.

Dedication

Sarah, my dear beloved: I cannot express sufficiently how much your love means to me. I think and I wonder but I am not able to imagine how I would be able to live let alone progress in life if you have not been in it. This is just but a gesture, a simple nod to, and a token of my love for all you have given me.

This is for you.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background

To believe Sir Tim Berners-Lee, the inventor of World Wide Web as we know it today, the future of the web – often called Web 3.0 – is and should be the Semantic Web [1]. In his TED talk [2] in 2009, he appealed to the people of the world to make Semantic Web a reality by sharing their data on the web rather than just sharing their documents. The world has, since then, answered his call by providing a huge and ever-growing flow of open data, creating the largest free, publicly available, semantically tagged, inter-linked, and machine-readable collection of structured data on the Internet called Linked Data [3, 4].

Linked Data is the heart of the Semantic Web. Although the two terms are often used interchangeably, the relationship between Linked Data and the Semantic Web is a matter of different opinions. While most people view the Semantic Web as the whole and Linked Data as the parts making up the whole, some believe Linked Data is the appropriate implementation of the Semantic Web concepts. To quote Sir Berners-Lee again, "Linked Data is the Semantic Web done right" [5]. To be more specific, Linked Data consists of a collection of different data structures provided by different organization. These data, however, have been carefully tagged in a manner which has made them machine-readable and linked to other parts of the Linked Data collection. Such design has provided unique opportunities for both research and practical applications. This dissertation attempts to exploit the inner-most core of the Linked Data, Wikipedia, in order to demonstrate the possibilities created by Linked Data.

One of the earliest adaptations and transformations of the conventional Web into the Semantic Web is none other than Wikipedia [6]. Wikipedia is a free encyclopedia of general knowledge which has been created by the global collaboration of writers; it includes articles about a vast variety of knowledge items in multiple languages and is supported by the non-profit Wikimedia Foundation. While Wikipedia is written and structured for human use, it has been an early subject of transformation to its Semantic Web counterpart called DBpedia [7]. In January 2007, DBpedia was initially released by people from the Free University of Berlin and the University of Leipzig in collaboration with OpenLink Software Incorporated. Since then, DBpedia has become the center of gravity for all other databases that join the Semantic Web community. DBpedia has converted the unstructured (or semi-structured) Wikipedia articles into the Semantic Web structured ontology standard called Resource Description Framework (RDF) [8]. RDF can simply be represented as a list of *subject*, *predicate*, and *object* triples. These triples, as will be shown later on, form a graph of knowledge items which are semantically connected to one another. It is these RDF graphs and certain sub-graphs which are the subject of attention and analysis of the presented work here.

The goal of this dissertation is to highlight the resources available from Linked Data, specifically DBpedia. The work presented here attempts to demonstrate how the semantic graph of Wikipedia/DBpedia can be utilized and depict certain applications resulting from the efforts. The main focus of the work is to introduce the graph-based analysis of semantic data and the solutions made possible by such analysis. Among the various solutions introduced, the entity detection and topic identification applications are the experimental venue which this work ventures into exploring more deeply and provides a comprehensive analysis of their affecting parameters.

1.2 Motivations

The motivation behind this work is mainly driven by the expanding interest and a recent increase in research based on the provided resources of the Linked Data. Chapter 2 presents a list of literature work on the topics of the Semantic Web, Linked Data, and their applications in text analytics. Specifically, it focuses on the use of Wikipedia in topic identification, the main theme of this dissertation. The interesting notion, however, is that while the official introduction of the concepts of Semantic Web [1] and the launch of Wikipedia were both initiated in 2001, it was only recently that researchers paid particular attention to the uses of Wikipedia in semantic challenges. Shortly after the first formulation, of how Linked Data should be designed, was presented in 2006 [3] and the launch of DBpedia in 2007, the research work using Wikipedia in text analytics started to show up more and more in literature and the number of publications has kept growing to this date. This is a positive indicator of the value of Linked Data, how it can produce opportunities that were previously impossible (or nearly so), and the effect it has brought to the text analytics community.

Another factor in motivating the author to pursue the presented topic is the project that jumpstarted the research. In early 2007, the Speech and Concept Understanding research team at the University of Waterloo’s Centre for Pattern Analysis and Machine Intelligence (UW CPAMI) undertook a project called *Parla Search Engine*. The goal of the project was to design a smart search engine for multimedia content. The Parla project consisted of multiple components, from automatic speech recognition to information retrieval. The component which started the research of this dissertation was called *Web Query Categorization* (WQC) or *Query Topic Identification* (QTI). The WQC target was to recognize the user's topic of interest based on his query, filter out the search engine results, and tailor it to match that topic. The exact details of the project and WQC component are explained later on in Chapter 3.

Finally, venturing into open problems from a novel point of view and a different direction along with scientific curiosity were most certainly forces behind motivating the research. There is a wide range of applications and problems to which the methods presented in this work can be applied. Some of these problems have been offered solutions from other directions, such as statistical approaches, and some would still need more research work. The new ground opened up by Linked Data resources is a tempting opportunity for innovation and the candidate has tried to take advantage of a portion of these resources hoping to provide a framework for future work and development of further innovative approaches.

1.3 Problem Domain and Contributions

In very generic terms, the scope of the dissertation is providing graph-based analysis of various text analytics topics using Linked Data resources. Since such a domain can be quite extensive, a certain core methodology is showcased which can easily be adapted to different applications and problems. In Chapter 3, some of these applications are described and, subsequently, a possible adaptation of the core methodology is presented as an example of how to tackle each of these specific problems. While being flexible to manage those applications, the presented methodology contains a number of key ingredients which are relied upon in each and every case. The main contribution of the thesis is to introduce a mentality blueprint of how to use these key ingredients to create proper solutions for various problems in this domain. In other words, the main contribution of the thesis is that it introduces a number of tools and demonstrates how these tools can be coupled together, albeit to different degrees in various cases, to tackle different problems.

The three key ingredients of the presented methodology are:

1. Semantic Graph Knowledgebase
2. Entity Detection
3. Semantic Analysis

These three ingredients are the general pillars of the proposed methodology. Each of them, however, has a number of sub-domains which are the tunable components making the methodology flexible to fit diverse problems. In Chapter 3, each of these components is explained in details and proper discussion is followed on how and when to use each component to best effect. These discussions constitute another contribution of this dissertation.

Following the description of methodology components, a list of the possible uses of the presented framework is offered. These applications are a small portion of possibilities brought upon by the ingredients mentioned above. Nevertheless, these applications render a sufficiently complete picture of what the problem domain is and how it can be undertaken. For each application, a possible solution is offered. The solution, however, is neither unique nor ideal as there are often too many parameters for each problem to consider. In order to demonstrate how these parameters affect the problem domain and how the solution should be evolved to match the problem, one specific application, *Query Topic Identification*, is studied in greater detail. The application has been analyzed thoroughly and the effects of each important factor have been examined. The experimental results of this study are then presented in Chapter 4. As a secondary problem, a small but intrinsically different application, *Speech Recognition*, is also introduced, studied, and empirically presented. These two experimental studies aim to showcase how the methodology can contribute to real-world problems. It is noteworthy to consider that both of these problems have been previously tackled by other methods; therefore, the methodology of this dissertation grants new perspectives and brings about balance by showing how the Semantic Web and Linked Data can complement the existing methods in domains which historically were handled by other means.

1.4 Organization of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 presents a review of the literature related to the scope of this thesis. It starts by presenting a background into the Semantic Web and its various technologies. Then, a description of Linked Data is provided along with guidelines on how to migrate to Linked Data as a realization of the Semantic Web. Finally, a brief look into Wikipedia uses in various problems pertaining to the topic of this thesis is presented.

Chapter 3 portrays the core methodology of this dissertation. It starts by explaining the structure of Wikipedia and DBpedia. Subsequently, it will describe the three key ingredients of the framework mentioned earlier, namely: Semantic Graph, Entity Detection, and Semantic Analysis. For each of these components, a number of sub-components are introduced and their uses are justified. Finally, a number of applications using the aforementioned framework are presented and a potential solution for each application is discussed. The first of these applications is the *Parla* project introduced earlier which motivated and initiated this thesis.

Chapter 4 demonstrates the experimental results of two studies in greater details: Query Topic Identification and Speech Recognition. Lastly, Chapter 5 will summarize and conclude the dissertation and provides some insight into possible future research on the topic.

Chapter 2
Background and Literature Review

2.1 Semantic Web

The methodology and framework proposed in this work heavily rely upon the Semantic Web framework and Linked Data resources. Thus, it is proper to briefly introduce the technology and provide some essential details used in this work. The Semantic Web, initiated and overseen by Tim Berners-Lee, is a collaborative movement led by the international standards body, the World Wide Web Consortium (W3C) [1]. As stated by W3C, Semantic Web mainly revolves around two issues: "common formats for integration and combination of data drawn from diverse sources" and "the language for recording how the data relates to real world objects" [9]. The goal of this movement is to convert the current state of the web, which mostly consists of unstructured or semi-structured documents, to a web of structured, semantically-tagged data. In order to achieve this task, a certain set of standards are proposed by the Semantic Web. The following provides a brief look inside the structure of the Semantic Web and some of its important components.

2.1.1 General Structure

In general, Semantic Web has a multi-layered architecture which consists of the data accompanied by semantics and reasoning tools. In order to make the Semantic Web work properly, web developers should be able to: have a common language to communicate and share their data, provide meaning and context for their data, and provide their reasoning schemes for further processing of their data. These three requirements are the reasons behind many of the layers designed for the Semantic Web. There are a few other layers, as well, which are mostly related to the front-end areas, such as user interface and applications. Figure 2.1 (a) shows the most famous structure for Semantic Web, originally presented in Tim Berners-Lee's talk in 2002, called the *layer cake* and has since evolved to cover new languages and components. Figure 2.1 (b) shows the updated version of the structure which has remained more or less the same since 2007.

2.1.2 URI/IRI

The Uniform Resource Identifier (URI) is the main and fundamental platform for the Semantic Web. URIs, as the name suggests, are strings of characters that identify resources over a network, mainly the World Wide Web. URIs have the capacity to be dereferenced and allow for the unambiguous identification of resources. All web page Uniform Resource Locators (URLs) are examples of URIs. However, URIs can be used for defining other concepts beside location of web pages. For instance, the URI `<http://www.w3.org/2002/07/owl#Thing>` defines the concept "Thing" which is the most general concept used in semantic hierarchies. Some developers provide corresponding URLs for the URIs they define and the web page pointed by the corresponding URL provides human-readable information about the URI. For instance, `<http://dbpedia.org/page/Life>` is a URL for a page which provides further information about the resource known by URI: `<http://dbpedia.org/resource/Life>`.

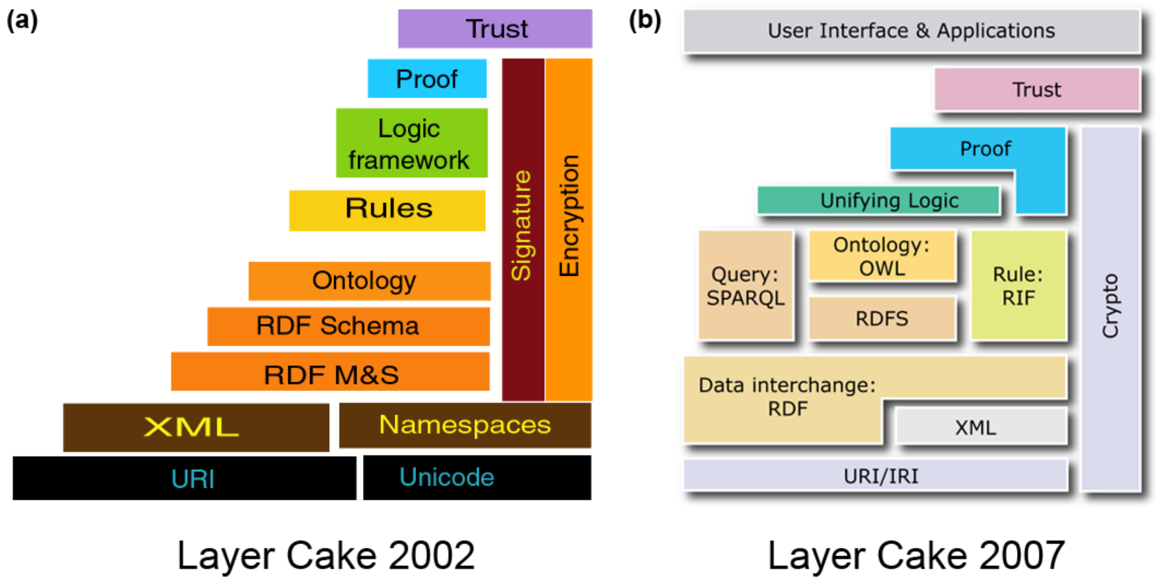


Figure 2.1: The layer cake [10, 11].

The upper level layers of the Semantic Web, specifically RDF, use URIs in their formats. The upper layers also incorporated Unicode to accommodate different character sets. However, since Internationalized Resource Identifiers (IRI), the latest extension of URIs, permit the use of full Unicode [12], the Unicode has been removed from latest version of the structure.

2.1.3 XML

The basic need for web developers in creating Semantic Web is data exchange and communication. Fortunately, because of XML [13] and its global approval from the web community, all developers agreed to use it as the common language to exchange data. The interoperability of XML provided a framework for developers to free themselves from previous rigid formats and to be able to express the semantics and their relationships much more easily [14]. As a result, all the other languages used in Semantic Web are built on top of XML [15]; as such these languages use XML syntax making XML a meta-language [16]. Moreover, XML Schema and Namespaces provide the capability of defining categories and structures within XML itself [16].

2.1.4 RDF

While XML provides a very flexible syntax for data exchange and XML Schema defines the data structure, neither can provide the framework capabilities which RDF offers: expressing semantics for the data. Designed similar to the classical conceptual modeling techniques such as entity-relationship or class diagrams, the W3C recommended language for expressing semantics is RDF [8]. According to W3C timeline, the first draft of RDF was released back in October 1997 [17]. W3C released the final version of the RDF in 2004 as a recommendation called RDF primer [18]. There is also an interest group called Semantic Web Interest Group [19], formerly known as RDF Interest Group [20], which serves as a public forum to discuss the use and development of the Semantic Web concepts including RDF. The final specifications for RDF can be found in a set of six documents called primer, syntax, semantics, vocabulary, concepts and test cases, found at [21].

The basic structure of RDF is rather simple: It merely consists of a set of triplets in the form of (*Subject*, *Predicate*, *Object*). Subject and Predicate are resources defined by URIs while Object can be either a URI or a literal [14]. This simple structure creates a set of entries which are, in fact, presentable as a directed graph called a *semantic network*. Figure 2.2 shows such a network for a simple example adopted from RDF Primer [18].

In this example, the RDF describes a resource which is about a certain person identified by the URI: `<http://www.w3.org/People/EM/contact#me>` whose name is “Eric Miller”, email address is `em@w3.org`, and title is “Dr.”. In this scenario, the resource URI indicated above is the subject. “Eric Miller”, `em@w3.org`, and “Dr.” are the objects. The predicates for these objects are respectively: “whose name is”, “whose email address is”, and “whose title is”. However, these predicates should be presented in URI form. The following shows the URIs for each predicate:

- "whose name is" as `<http://www.w3.org/2000/10/swap/pim/contact#fullName>`
- "whose email address is" as `<http://www.w3.org/2000/10/swap/pim/contact#mailbox>`
- "whose title is" as `<http://www.w3.org/2000/10/swap/pim/contact#personalTitle>`

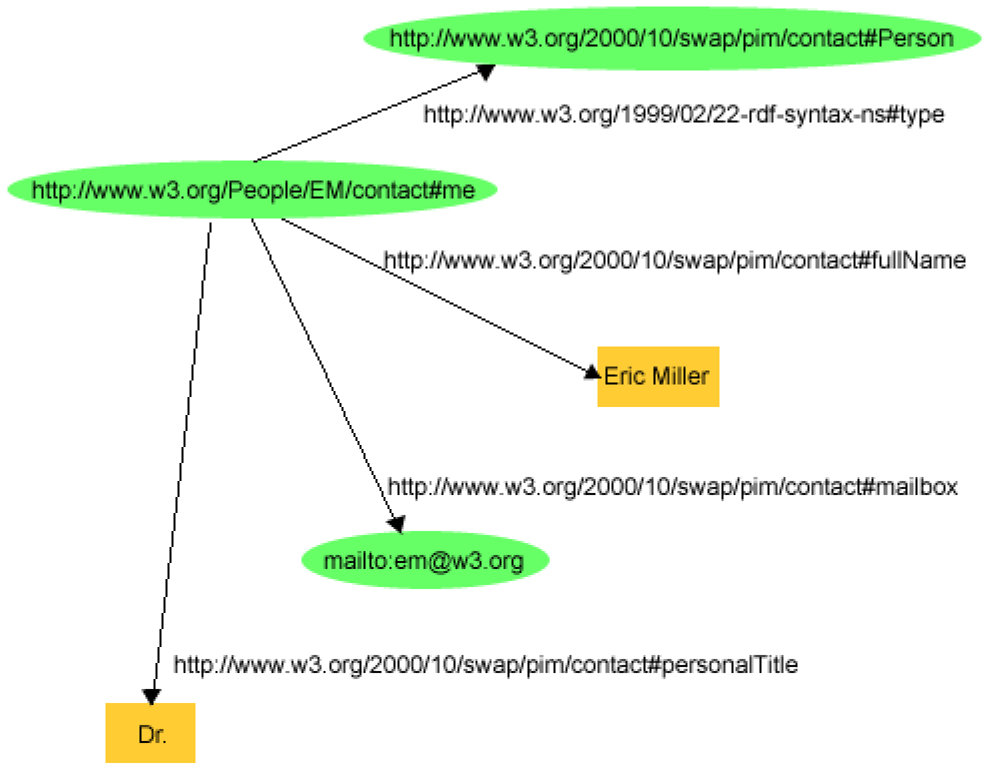
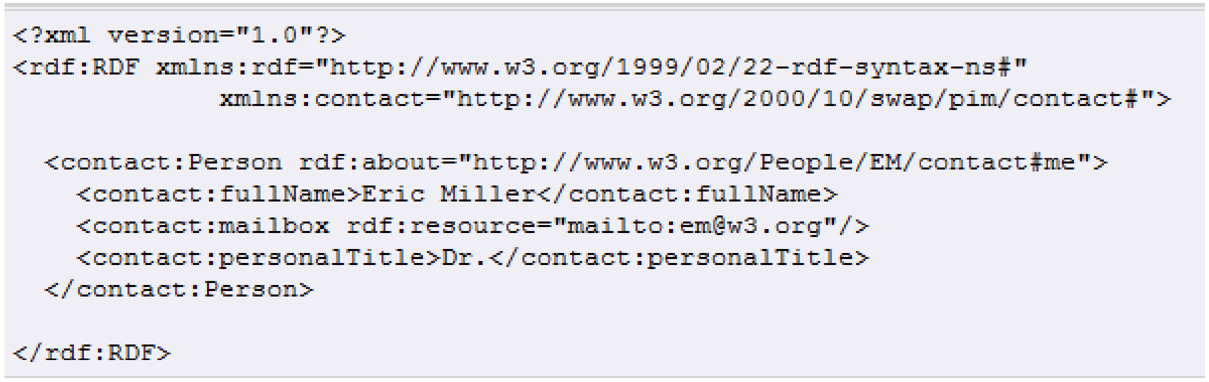

http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Eric Miller
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:em@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Dr.

Figure 2.2: A sample RDF graph (semantic network).

There is also another relationship which indicates that the subject being described in this example has a type defined by URI: `<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>` and the type is a Person defined by URI: `<http://www.w3.org/2000/10/swap/pim/contact#Person>`. Note that the definition of the type as the predicate is a standard RDF URI which means it can be unambiguously parsed by RDF parsers; however, the definition of a Person, for instance, uses a local semantic RDF definition. Such local RDF descriptions can be further refined by the provider of this RDF sample using predicates that equate the local definition of the person with a globally known URI.

Figure 2.3 shows the RDF code corresponding to this example. Note that in this code, XML namespaces are used to shorten the common URI prefixes. For instance, the namespace `xmlns:contact` is defined as `<http://www.w3.org/2000/10/swap/pim/contact#>` which means that `contact:Person` is equal to `<http://www.w3.org/2000/10/swap/pim/contact#Person>`. Also note that using `contact:Person` as the starting tag implies the type relation in RDF parser rules and, therefore, the type association can be omitted.



```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>

</rdf:RDF>
```

Figure 2.3: A sample RDF code.

Finally, the above RDF example can be described in triple format. The triple format is a simple and commonly used method of expressing RDF data. Table 2.1 shows four triples, each accounting for an edge in the RDF graph of Figure 2.2. When recorded in a file, each triple row is terminated by a blank space character followed by a period.

Table 2.1: A sample RDF description in triples format.

Subject	**Predicate**	**Object**
`<http://www.w3.org/People/EM/contact#me>`	`rdf:type`	`contact:Person`
`<http://www.w3.org/People/EM/contact#me>`	`contact:fullName`	“Eric Miller”
`<http://www.w3.org/People/EM/contact#me>`	`contact:mailbox`	`mailto:em@w3.org`
`<http://www.w3.org/People/EM/contact#me>`	`contact:personalTitle`	“Dr.”

In the above table, the subject URI is presented in its full form. However, the other columns use namespaces to indicate the URIs (note that mailto: is not a namespace, but rather an identifier such as http:). Table 2.2 lists a number of namespaces frequently used in Semantic Web community for resource definition along with a few namespaces for largely used data sources such as DBpedia.

2.1.5 Ontology

Providing semantics for data is not enough to guarantee data is understandable by various parties. Each party may use different identifiers or resources to describe a single term [1]. In order to complement the semantic circle, each party has to provide a vocabulary of terms and a set of inference rules so that other parties can comprehend the semantics used and, if necessary, deduct new semantics using the set of rules. This collection of vocabulary and rules is called ontology.

Table 2.2: Common namespace prefixes and corresponding URIs.

Namespace Prefix	Namespace URI
`rdf:`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
`rdfs:`	`http://www.w3.org/2000/01/rdf-schema#`
`owl:`	`http://www.w3.org/2002/07/owl#`
`dc:`	`http://purl.org/dc/elements/1.1/`
`dcterms:`	`http://purl.org/dc/terms/`
`skos:`	`http://www.w3.org/2004/02/skos/core#`
`foaf:`	`http://xmlns.com/foaf/0.1/`
`dbpedia:`	`http://dbpedia.org/resource/`
`dbpedia-owl:`	`http://dbpedia.org/ontology/`

There exist quite a few ontology languages which were developed by different organizations. Among them SHOE [22] was a language based on HTML presented in 2000. XOL [23] was one of the early languages built using XML. After the introduction of RDF, W3C further introduced RDF Schema, also known as RDFS, as an extension to RDF. Similar to XML Schema, RDFS provided the structure definition for RDF which, in fact, acted as a kind of ontology. RDFS, which is very similar to object-oriented languages, enabled developers to express classes, subclasses, their relations, and properties of each class such as domain and range [24]. DAML-ONT [25] is another ontology language developed by DARPA. DAML-ONT covered the basic concepts similar to RDFS as well as inference logic. OIL [26] was a European counterpart ontology language and its foundation was very similar to RDFS, but it was based on description logic. DAML+OIL [27] was a descendant of the previous two languages and tried to include the expressiveness of both languages. Finally, OWL, or the Web Ontology Language [28], was introduced as the W3C ontology language of choice in 2004. OWL is a successor of DAML+OIL and has three different levels: Lite, DL, and Full. These three differ in the degree of expressiveness and completeness. OWL is now experiencing its second version called OWL 2 released in 2009 [29]. This new version is an extension and revision of OWL and, therefore, is fundamentally very similar but adds new functionality with respect to OWL. Figure 2.4 shows the results of a survey assessing the amount each ontology language is used among different types of researchers [30]. OWL is the most common ontology among developers, as shown in this figure. OWL uses RDF notations but it provides more features including cardinality, transitive property, disjoint union of classes, and richer data types.

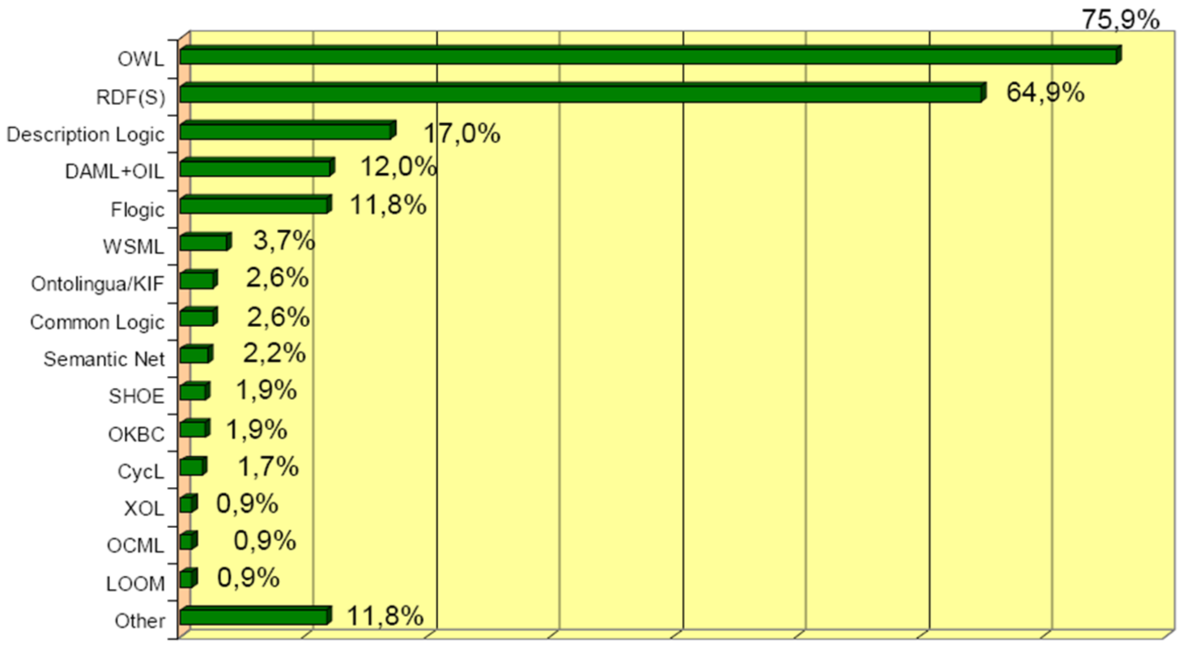
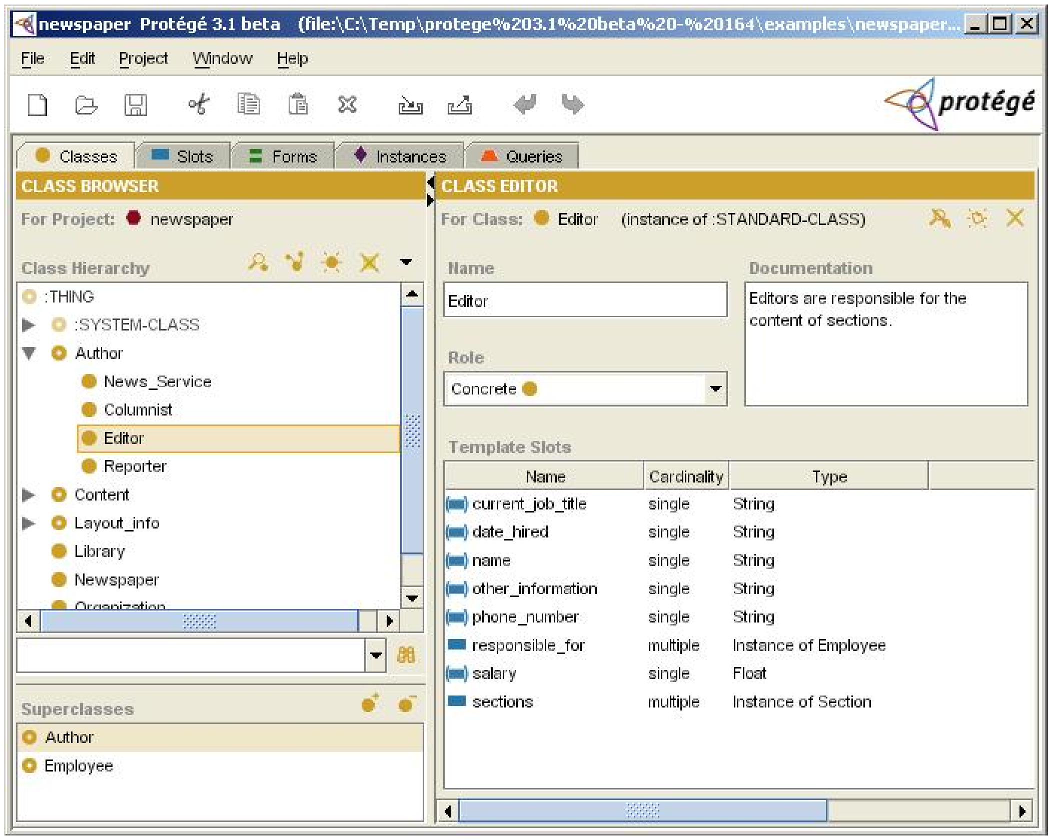


Figure 2.4: Ontology languages and their use by researchers (adopted from [30]).

2.1.6 Ontology Editors

Ontology editors are tools created to facilitate the implementation, organization, and modification of custom ontology description data for web developers. These editors essentially provide basic levels of ontology development, such as defining classes, editing class properties, specifying relationships, and imposing constraints, and may be able to provide syntax checking, as well [16]. Some powerful tools can also combine different ontology descriptions. The best available editor is called Protégé [31]. It is a Java-based, open-source editor developed by Stanford University that supports RDF(S) and OWL. Figure 2.5 provides a screenshot of this editor.

There are two other famous ontology editors: SWOOP [32, 33] and OntoEdit [34]. Developed in University of Maryland, SWOOP is also an open-source, Java-based editor which acts as an OWL ontology web browser/editor. It combines the power of Protégé with a web-based environment called Ontolingua to provide an easy-to-use browser and editor [35]. OntoEdit, now commercially available as Ontostudio, used to be a web service developed in University of Karlsruhe which supported F-Logic, RDFS, and OIL. Figure 2.6 shows the results of the survey (similar to Figure 2.4) assessing the percentage usage of different ontology editors.



For Project: newspaper

:THING
:SYSTEM-CLASS
Author
News_Service
Columnist
Editor
Reporter
Content
Layout_info
Library
Newspaper
Organization

Superclasses: Author, Employee

For Class: Editor (instance of :STANDARD-CLASS)

Name: Editor

Documentation: Editors are responsible for the content of sections.

Role: Concrete

Name	Cardinality	Type
current_job_title	single	String
date_hired	single	String
name	single	String
other_information	single	String
phone_number	single	String
responsible_for	multiple	Instance of Employee
salary	single	Float
sections	multiple	Instance of Section

Figure 2.5: A screenshot from Protégé (adopted from [31]).

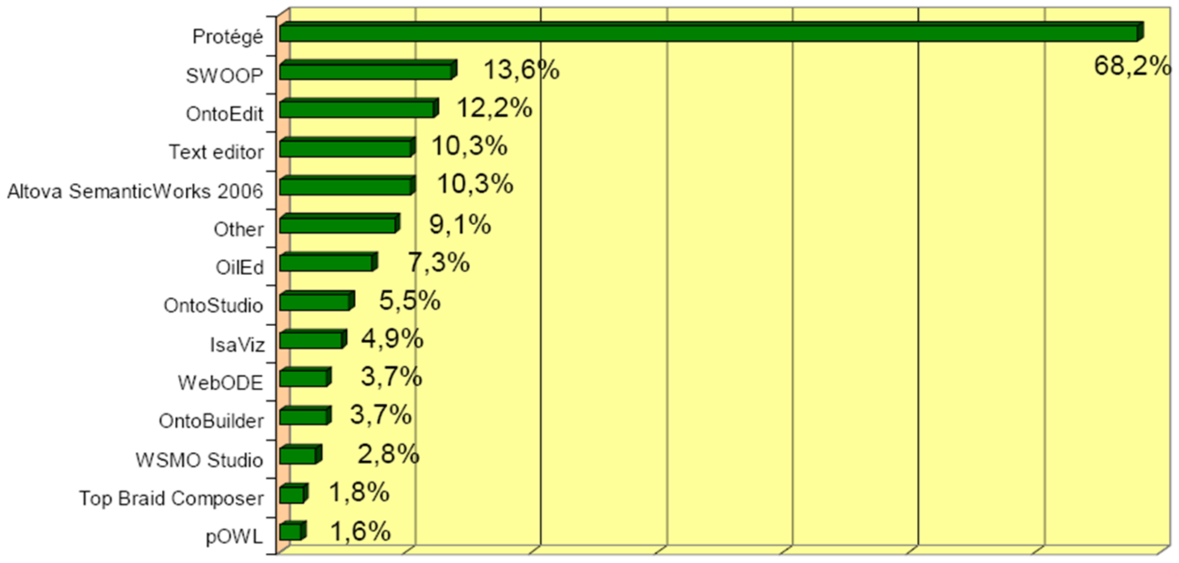


Figure 2.6: Ontology editors and their use by researchers (adopted from [30]).

2.1.7 Reasoning Engines

Reasoning engines, or semantic reasoners, are tools which use the provided facts and inference rules of a certain ontology dataset to deduce new facts about that ontology. The newly deducted facts help to answer queries which could not originally be answered using initial facts. Having been studied by the Artificial Intelligence (AI) community for quite some time, a number of reasoning engines were developed to support ontology standards discussed previously.

One of the most popular reasoning engines is called Jena [36, 37] which was originally developed by HP Labs at Bristol in 2000. Jena was an open-source project and once HP stopped supporting it in 2009, the Apache Software Foundation adopted the project following the application of the project team in November 2010. Jena is more than an inference engine; it is in fact, a framework for Semantic Web application development. It includes a number of components to handle RDF, ontology, and the SPARQL query language [38], a Semantic Web query language similar to relational database management systems query languages like SQL. Furthermore, Jena's reasoning engine supports RDFS and OWL-DL entailment.

The two other popular reasoning engines are Racer [39, 40], which is based on description logic and supports RDFS, DAML and OWL, and Pellet [41, 42], which is made by the same people responsible for SWOOP and works with SWOOP as an ontology consistency analyzer. Figure 2.7 lists different existing reasoners and shows how popular they are.

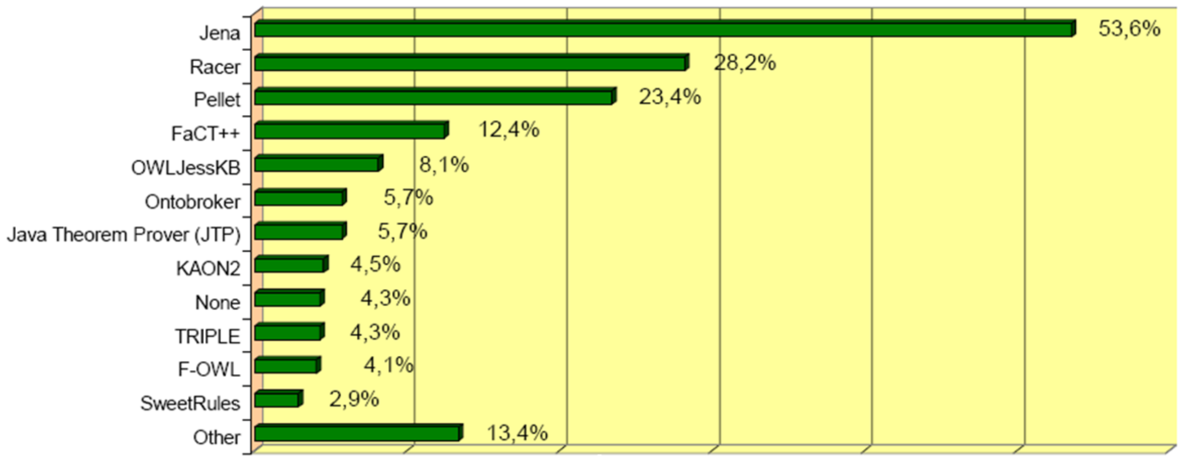


Figure 2.7: Reasoners and their use by researchers (adopted from [30]).

2.2 Linked Data

Linked Data is the realization of the Semantic Web. The Semantic Web is a “Web of Data” of any type imaginable: people, events, places, organizations, and so on. The Semantic Web includes a collection of technologies (RDF, OWL, SKOS, SPARQL, etc.) which offer an environment where a specific application is able to request certain data inquires or, using the provided ontology, infer new facts. However, to have an actual web of data, it is imperative to convert the huge amount of data on the Web to “a standard format, reachable, and manageable by Semantic Web tools” [43]. Moreover, it is not enough to only provide access to data. The data providers should also offer the semantic relationships among data. Otherwise, the Web of Data will be reduced to a sheer collection of datasets. Such collection of interrelated datasets on the Web is referred to as Linked Data [43].

The implementation of Linked Data requires technologies to be offered under the umbrella of a common format (RDF) so that the existing databases in various formats, such as relational, XML, or HTML, can be either converted or accessed on-the-fly. Another important requirement is to provide query endpoints to facilitate data access. While W3C offers a variety of data access technologies (RDF, GRDDL, POWDER, RDFa, R2RML, RIF, SPARQL), there was a motivational impasse in the realization of Linked Data.

In order to encourage data providers to convert and provide access to their datasets using Semantic Web technologies, a few interesting applications or frameworks were needed as incentives. On the other hand, to create such incentives, large enough datasets in Semantic Web format should have been available. To resolve the deadlock, some organizations decided to bootstrap the Linked Data by unilaterally generating large Semantic Web datasets and creating a community called Linking Open Data (LOD) [44].

One such large Linked Dataset generated by LOD is DBpedia. This dataset, which is at the heart of Linked Data, offers the content of Wikipedia in RDF format. DBpedia is particularly significant not only because of providing access to Wikipedia data, but also due to integration of links to other datasets on the Web, e.g. GeoNames [45]. DBpedia also provides “same-as” RDF triples which equate the URI of certain items in its knowledgebase with the URI of equivalent items in other similar datasets, such as YAGO [46, 47]. Providing these extra links enables applications to exploit and integrate more knowledge and facts from several datasets which may improve the user experience. Figure 2.8 shows one of the early stages of LOD in May 2007 with only 12 datasets and Figure 2.9

presents its latest state as of September 2011 with 295 datasets. The comparison demonstrates the growing interest in joining the Linked Data movement by various organizations all over the world.

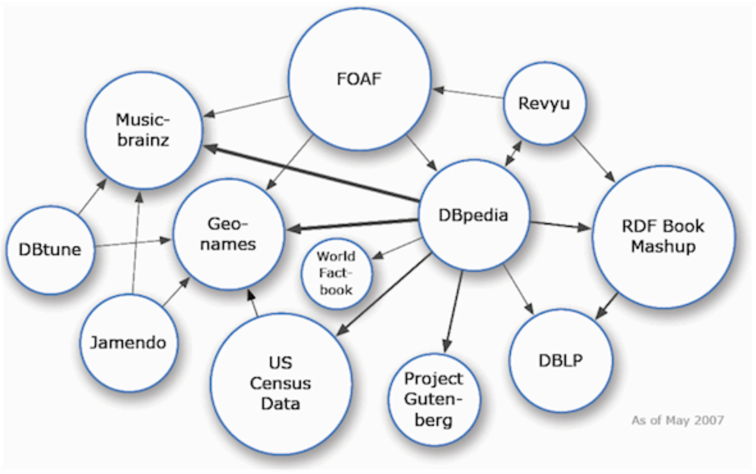


Figure 2.8: LOD cloud diagram in May 2007 (adopted from [48]).

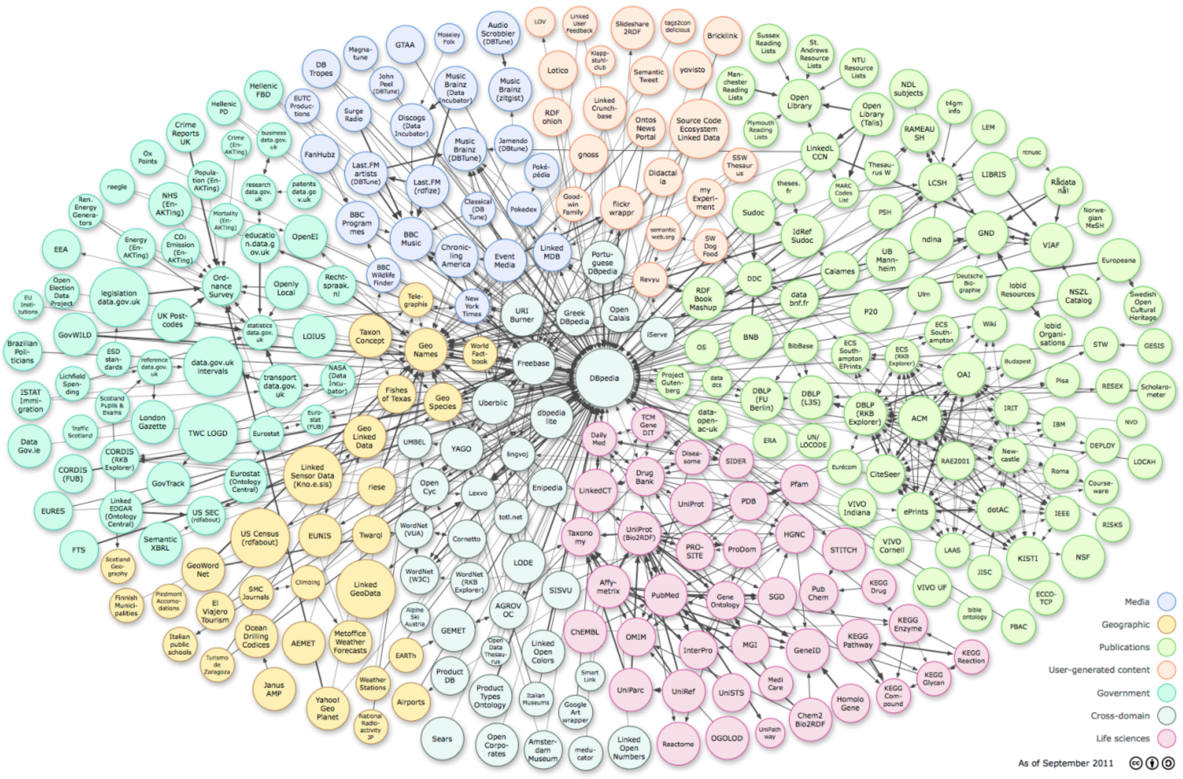


Figure 2.9: LOD cloud diagram as of September 2011 (adopted from [48]).

In order to generate a dataset in the proper Linked Data format, Tim Berners-Lee laid out a few rules in his notes on Linked Data design issues [3]. The following reiterates these rules, which are commonly known as the 'Linked Data principles'.

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things

The first rule is trivial since there would be no Semantic Web in the absence of URIs as mentioned earlier in layer cake architecture. The second rule is also globally understood given the available HTTP infrastructure of web pages which would be convenient landing places for description documents. An example was given earlier about the DBpedia URI and corresponding URL for the 'Life' resource. The third rule suggests that the URIs should come with information which can be used to explore and exploit relationships and rules. Such information should be in a standard Semantic Web format to facilitate access. Finally, the fourth rule again emphasizes including connections to other datasets, a rule which is derived from the basic nature of Linked Data.

Finally, Berners-Lee suggests a star rating scheme to encourage government data owners and other organizations to provide data as Linked Open Data rather than Linked Data. The four rules above can be used as guidelines for creating Linked Data. However, if one wants to have Linked Open Data, the data should be released under an open license such as Creative Commons CC-BY [49]. Table 2.3 presents this scheme which makes data progressively more powerful and useful as it gets more stars.

Table 2.3: Linked Open Data star rating system (adopted from [3]).

★	Available on the web (whatever format) *but with an open license, to be Open Data*
★★	Available as machine-readable structured data (e.g. excel instead of image scan of a table)
★★★	Same as (2) plus non-proprietary format (e.g. CSV instead of excel)
★★★★	All the above plus: Use open standards from W3C (RDF and SPARQL) to identify things, so that people can point at your stuff
★★★★★	All the above, plus: Link your data to other people's data to provide context

2.3 Wikipedia

The foundation of the methodology presented in this dissertation is undoubtedly Wikipedia. While a complete description of the building blocks of Wikipedia is presented in Chapter 3, it is fitting to provide a brief background on the uses of Wikipedia in a few fields related to this work. The following sections provide a few examples of Wikipedia application in literature. First, a quick look at the connections between Wikipedia and the Semantic Web is presented. Afterwards, a background on topic identification, which is an important portion of this dissertation, is introduced. As presented in Chapter 4, query topic identification is the dominant application for which the experimental results have been discussed; therefore, the review is more focused on query topic identification. However, the last section provides an insight into the application of Wikipedia in a more general field: text classification.

2.3.1 Wikipedia and Semantic Web

The idea of integrating Semantic Web concepts and Wikipedia/Wikis has been the subject of discussion for a while now. Besides DBpedia, there are a number of other works done on the subject. Völkel *et al.*, for example, proposes that semantics will be added to Wikipedia articles in the form of category, typed links, and attributes [50]. The paper states that categories already are used in Wikipedia and by adding simple prefixes to internal Wiki links, articles would be linked semantically. Attributes provide literals (strings, numbers, etc. which are not specific articles) to the articles. In a way, typed links resemble predicates and attributes resemble objects in RDF. Furthermore, the authors are affiliated with Karlsruhe Institute of Technology which was behind the development of the Semantic MediaWiki (SMW) [51, 52]. SMW is an extension of MediaWiki, the software which powers Wikipedia. This extension is free and open-source which provides the capability of storing and querying data within the Wiki pages. This creates a framework for turning Wikis into powerful and flexible collaborative databases. Moreover, the data entered into these Wiki pages can effortlessly be converted into Semantic Web format.

2.3.2 Wikipedia and Topic Identification

Query topic identification, or query classification, is the task of NLP that focuses on inferring the domain information surrounding user-written queries and on assigning to each query the best category label from a predefined set. Given the ubiquity of search engines and question-handling systems today, this challenge has been receiving a growing amount of attention. For example, it was the topic of the ACM’s annual KDD CUP competition in 2005 [53], where 37 systems competed to classify a set of 800 real web queries into a set of 67 categories designed to cover most topics found on the internet. The winning system was designed to classify a query by comparing its word vector to that of each website in a set pre-classified in the Google directory. The query was assigned the category of the most similar website, and the directory’s set of categories was mapped to the KDD CUP’s set [54]. This system was later improved by introducing a bridging classifier and an intermediate-level category taxonomy [55, 56].

Most query classifiers in the literature, like the system described above, are based on the idea of mapping the queries into an external knowledge source (an objective third-party knowledgebase) or internal knowledge source (user-specific information) to classify them. This simple idea leads to a great variety of classification systems. Using an internal knowledge source, Cao *et al*. [57] developed a query classifier that disambiguates the queries based on the context of the user’s recent online history. On the other hand, many very different knowledge sources have been used in practice, including ontology [58], websites [59], web query logs [60], and Wikipedia [61, 62].

Exploiting Wikipedia as a knowledge source has become commonplace in scientific research. Several hundreds of journal and conference papers have been published using this tool since its creation in 2001. However, while both query classification and NLP using Wikipedia are common challenges, there have not been many query classification systems based on Wikipedia. One such system was proposed by Hu *et al*. [61]. Their system begins with a set of seed concepts to recognize, and it retrieves the Wikipedia articles and categories relevant to these concepts. It then builds a domain graph by following the links in these articles using a Markov random walk algorithm. Each step from one concept to the next on the graph is assigned a transition probability, and these probabilities are then used to compute the likelihood of each domain. Once the knowledgebase has been built in this way, a new user query can be classified simply by using its keywords to retrieve a list of relevant Wikipedia domains, and sorting them by likelihood. Unfortunately, their system remained small-scale and limited to only three basic domains, namely “travel”, “personal name” and “job”. It is not a general-domain classifier such as the one introduced in this dissertation.

Another query classification system was designed by Khoury [62, 63]. It follows Wikipedia’s encyclopedia structure to classify queries step-by-step: first, using the query’s words to select titles, second, selecting articles based on these titles, and third, selecting categories from the articles. At each step, the weights of the selected elements are computed based on the relevant elements in the previous step: a title's weight depends on the words that selected it, an article's weight on the titles', and a category's weight on the articles'. Unlike [61], this system was a general classifier that could handle queries from any domain.

2.3.3 Wikipedia and Text Classification

While using Wikipedia for query classification has not been a common task, there have been several document classification projects done using that resource which are worth mentioning. Schönhofen [64] successfully developed a complete document classifier using Wikipedia by mapping the document's vocabulary to titles, articles, and finally categories, and weighting the mapping at each step. Alternatively, other authors use Wikipedia to enrich existing text classifiers by improving upon the simple bag-of-words approach. The authors of [65] use it to build a kernel to map the document's words to the Wikipedia article space and classify from there, while the authors of [66] and [67] use it for text enrichment: to expand the vocabulary of the text by adding relevant synonyms taken from Wikipedia titles. Interestingly, improvements are reported in the classification results of [64], [66] and [67], while only [65] reports worse results than the bag-of-words method. The conclusion seems to be that working in the word space is the better option, a conclusion that [65] also shares.

2.4 Summary

This chapter provided a background on the topics fundamental to the understanding of the methodology presented in the upcoming chapters. First, the Semantic Web was introduced and its various components were presented in the layer cake structure. Afterwards, each of these components was studied. Specifically, RDF was discussed in details as it is one of the most important technologies in Semantic Web standards. After introducing the Semantic Web, the concepts of Linked Data and Linked Open Data were introduced as the physical implementations of the Semantic Web principles. Lastly, a review of some of the uses of Wikipedia in literature, which are closely related to the topic of this dissertation, was presented. Among them, query topic identification was the key application as it is the target of the experiments of this dissertation. Finally, it is important to note that some of the works in related literature are mentioned in later chapters where applicable to the context.

Chapter 3
Semantic Analysis of Wikipedia

3.1 Introduction

In this chapter, the general methodology and its specific components are discussed. As both titles of the dissertation and this chapter suggest, the main building block of the methodology is Wikipedia. Therefore, Section 3.3 starts by explaining the structure of Wikipedia and digs into how it will be used to solve problems. Afterwards, the Semantic Web version of Wikipedia, called DBpedia, will be introduced. These two will constitute the foundation of the methodology: the knowledgebase that the framework is relied upon.

In Section 3.5, the semantic graph of the knowledgebase will be the topic of discussion. The semantic graph, one of the key ingredients mentioned in Chapter 1, is the most important part of the knowledgebase with respect to the problems presented in this work. Therefore, the focus is directed towards how the semantic graph connects *entities* and *topics* to each other and one another: entity-to-entity, topic-to-topic, and entity-to-topic.

Once the knowledgebase is fully discussed, Section 3.6 and Section 3.7 will present the two main algorithms of the methodology upon which all the offered solutions and experiments are based. These two algorithms, called entity detection and semantic analysis, are the other two key ingredients of the methodology. They will present two general ideas for how to tackle different problems. The specific algorithm used for different problems is usually a different variation of each of these algorithms tailored for that specific problem. However, the order of the application of these two algorithms is always the same. The entity detection should always be performed before semantic analysis since semantic analysis depends upon, and operates on, the output of entity detection.

The final part of this chapter will present a few interesting problems and suggest the potential solutions to these problems. These solutions are, as expected, based on the proper variation of two discussed algorithms using a proper portion of the knowledgebase.

3.2 Problem Definition

Before delving into the discussion of the building blocks of the methodology, it is helpful to describe the generic problem definition which the methodology is trying to address. While a crisp formulation cannot be achieved to cover all the use cases, Table 3.1 illustrates a generic problem definition which can be adapted to various specific problems.

Table 3.1: Generic problem definition.

Input	A textual object in any format e.g.: • Letters • Bag of words • Sequential words • Bag of sentences • Sequential sentences • Parallel sentences • Text document
Processing	Perform any number of the following tasks: • Detect potential entities (possibly ranked) • Traverse the semantic graph • Gather and rank semantically similar entities/topics • Determine high scored target entities/topics
Output	Suggest a ranked list or an iterative recommendation of one or a combination of the following items: • Entities • Topics • Part of the input • Any textual object

The input portion of the problem is some kind of textual object. As mentioned in Table 3.1, it can be any collection of words, sentences, letters, or even, in the case of speech recognition application, a set of parallel potential candidate sentences as we will observe. These collections can be ordered or unordered text. The size is also variable. It is possible that the input would be a short text like a web query or a large document such as an e-mail. The larger sized text objects are usually broken into smaller ones. The details of such pre-processing on input and handling of different types of input data are later discussed in Section 3.6.

The processing portion is usually the part that varies greatly from one application to another. The steps mentioned above are the ones taken in the provided solutions in Section 3.8. The processing portion needs a knowledgebase in order to perform the semantic analysis required for solving related problems. More specifically, the processing unit employs two main concepts of the knowledgebase called entities and topics. While Section 3.3 and Section 3.4 explain the main structure of the knowledgebase, Section 3.5 will give an in depth look into the concepts of entity and topic and how they are related in the semantic graph. Based on different applications, the solutions might be tipped towards one of these two concepts. For instance, the query topic identification problem uses more of the topic graph while speech recognition and context awareness applications are dealing mainly with entities. In any case, the detection of entities is always an included part of the algorithms presented here. Afterwards, it is possible to traverse the graph, find semantically similar entities or topics, and rank them based on a relevancy equation. In some cases, the targets are known, e.g. categories in web query categorization. Therefore, the processing unit will try to map the input to those target outputs. In other cases, such as text prediction and completion, there is no specific target available; this means the processing unit can only provide a list of candidate targets based on semantic similarity to the input. Sections 3.6 and 3.7 will explore the different aspects of the processing portion.

Once the processing is done, the output is presented usually as a ranked list of items or iterative recommendation of items. The items can be, as mentioned, entities, topics, or any type of textual object. The output may also be a combination of different items. For example, logical AND/OR combination of topics can be the result of a topic identification problem.

To summarize the problem definition given a text input, we would like to perform a semantic analysis of the input based on a knowledgebase of entities and topics and return the most relevant and closest desired target object or objects. Having had the general picture of problem definition in mind, we can now proceed to the description of the fundamental components of the methodology. Once all the components are introduced, Section 3.8 will show how the above generic problem definition is evolved into different specific applications and how the portions are adapted to each application.

The very first component to study is the knowledge platform upon which all the processing algorithms are based. As mentioned earlier, the knowledgebase mainly consists of entities, topics, and their relationships. Understanding these concepts is paramount to understanding the methodology. Therefore, in the next three sections, we will thoroughly explain the structure of the knowledgebase and two parts of the semantic graph related to entities and topics.

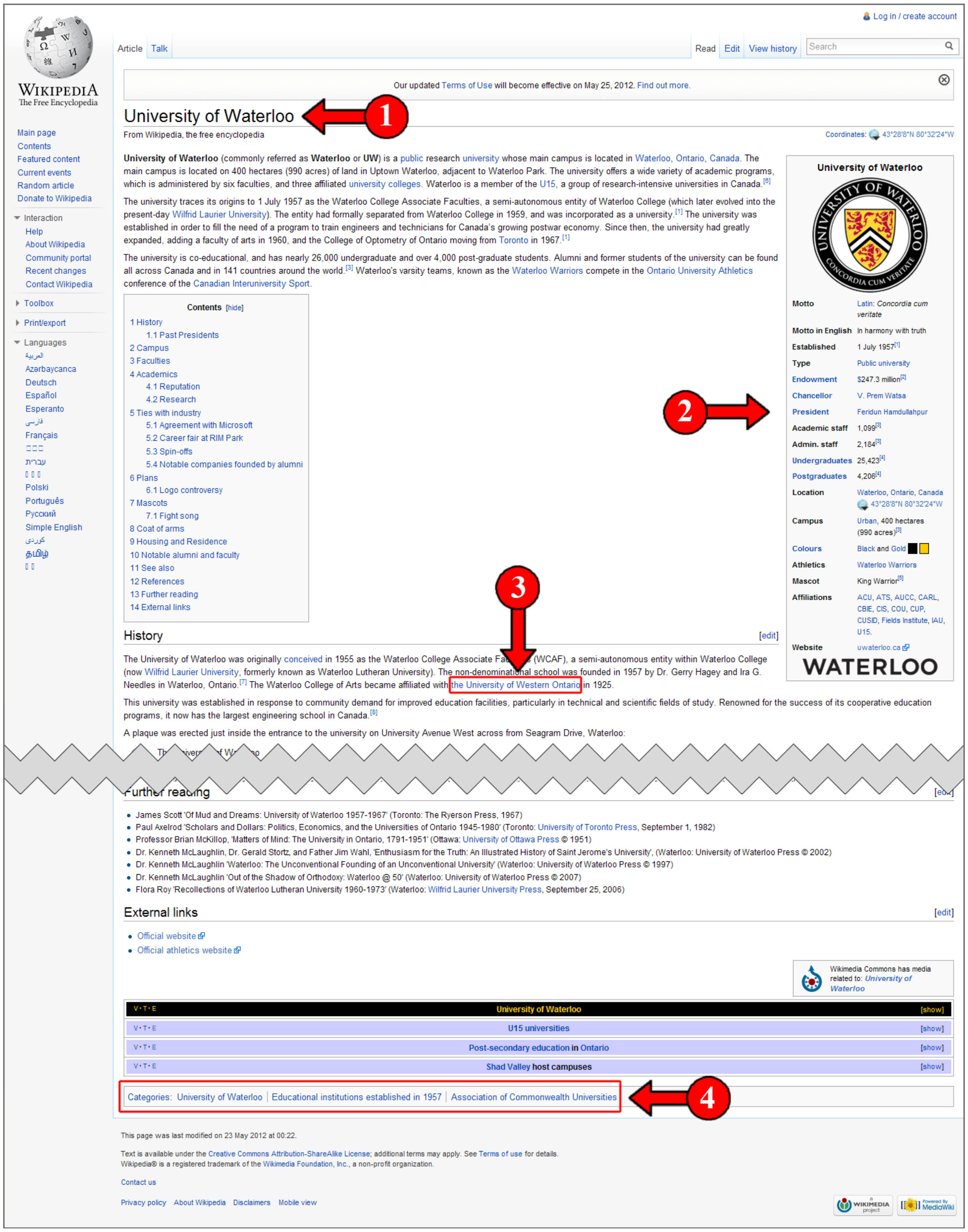
3.3 Wikipedia

The fundamental platform of the methodology is the knowledgebase over which all the presented algorithms are constructed. The choice of the knowledgebase for this work is Wikipedia. As mentioned before, Wikipedia has been a popular resource for numerous research works, some of which were mentioned in Section 2.3. It has also become the major source of open data for the improvement of the Semantic Web and Linked Data community [4, 68]. The main reason Wikipedia gets such attention is its sheer extent over a wide range of knowledge topics. According to the official Wikipedia statistics webpage [69], there have been about 4 million articles in the English branch of Wikipedia and more than 21 million articles in all languages of Wikipedia as of April 2012. This much volume opens up unique possibilities for research and makes Wikipedia the main player in the Linked Data community. In the following, the structure of Wikipedia is studied, its advantages and problems are discussed, and the process of converting Wikipedia to the semantic graph is explained.

Wikipedia can simply be defined as a collection of articles about various concepts and knowledge items. These articles are written in different languages through the collaboration of different authors. One of the unique characteristics of Wikipedia is the internal links between articles. These links are many-to-many connections which relate each article to many others. These links and the frequency of their existence are mainly decided by human intuition of authors and collaborative correction mechanism of Wikipedia. If certain parts of an article are missing, inadequate, and/or incorrect, the subsequent authors will try to fix or compensate them.

Figure 3.1 displays a snapshot of an article about the University of Waterloo. The contents of the middle of the page are removed and only the top and bottom of the page is depicted in this figure since these parts are the locations with the key information. Four items have been labeled on this figure. These items are the main elements of Wikipedia used to create the semantic graph. The following subsections will explain each of these elements:

1. Title of the article
2. Infobox
3. Wikilinks
4. Categories



Our updated Terms of Use will become effective on May 25, 2012. Find out more.

University of Waterloo

From Wikipedia, the free encyclopedia

Coordinates: 43°28′8″N 80°32′24″W

University of Waterloo (commonly referred as **Waterloo** or **UW**) is a public research university whose main campus is located in Waterloo, Ontario, Canada. The main campus is located on 400 hectares (990 acres) of land in Uptown Waterloo, adjacent to Waterloo Park. The university offers a wide variety of academic programs, which is administered by six faculties, and three affiliated university colleges. Waterloo is a member of the U15, a group of research-intensive universities in Canada.[6]

The university traces its origins to 1 July 1957 as the Waterloo College Associate Faculties, a semi-autonomous entity of Waterloo College (which later evolved into the present-day Wilfrid Laurier University). The entity had formally separated from Waterloo College in 1959, and was incorporated as a university.[1] The university was established in order to fill the need of a program to train engineers and technicians for Canada's growing postwar economy. Since then, the university had greatly expanded, adding a faculty of arts in 1960, and the College of Optometry of Ontario moving from Toronto in 1967.[1]

The university is co-educational, and has nearly 26,000 undergraduate and over 4,000 post-graduate students. Alumni and former students of the university can be found all across Canada and in 141 countries around the world.[3] Waterloo's varsity teams, known as the Waterloo Warriors compete in the Ontario University Athletics conference of the Canadian Interuniversity Sport.

Contents [hide]

1 History
1.1 Past Presidents
2 Campus
3 Faculties
4 Academics
4.1 Reputation
4.2 Research
5 Ties with industry
5.1 Agreement with Microsoft
5.2 Career fair at RIM Park
5.3 Spin-offs
5.4 Notable companies founded by alumni
6 Plans
6.1 Logo controversy
7 Mascots
7.1 Fight song
8 Coat of arms
9 Housing and Residence
10 Notable alumni and faculty
11 See also
12 References
13 Further reading
14 External links

University of Waterloo

Motto	Latin: *Concordia cum veritate*
Motto in English	In harmony with truth
Established	1 July 1957[1]
Type	Public university
Endowment	$247.3 million[2]
Chancellor	V. Prem Watsa
President	Feridun Hamdullahpur
Academic staff	1,099[3]
Admin. staff	2,184[3]
Undergraduates	25,423[4]
Postgraduates	4,206[4]
Location	Waterloo, Ontario, Canada 43°28′8″N 80°32′24″W
Campus	Urban, 400 hectares (990 acres)[3]
Colours	Black and Gold ■ 🟨
Athletics	Waterloo Warriors
Mascot	King Warrior[5]
Affiliations	ACU, ATS, AUCC, CARL, CBIE, CIS, COU, CUP, CUSID, Fields Institute, IAU, U15.
Website	uwaterloo.ca

WATERLOO

History [edit]

The University of Waterloo was originally conceived in 1955 as the Waterloo College Associate Fa s (WCAF), a semi-autonomous entity within Waterloo College (now Wilfrid Laurier University, formerly known as Waterloo Lutheran University). The non-denominational school was founded in 1957 by Dr. Gerry Hagey and Ira G. Needles in Waterloo, Ontario.[7] The Waterloo College of Arts became affiliated with the University of Western Ontario in 1925.

This university was established in response to community demand for improved education facilities, particularly in technical and scientific fields of study. Renowned for the success of its cooperative education programs, it now has the largest engineering school in Canada.[8]

A plaque was erected just inside the entrance to the university on University Avenue West across from Seagram Drive, Waterloo:

Th iver f W o

urther rea ng [ed]

- James Scott 'Of Mud and Dreams: University of Waterloo 1957-1967' (Toronto: The Ryerson Press, 1967)
- Paul Axelrod 'Scholars and Dollars: Politics, Economics, and the Universities of Ontario 1945-1980' (Toronto: University of Toronto Press, September 1, 1982)
- Professor Brian McKillop, 'Matters of Mind: The University in Ontario, 1791-1951' (Ottawa: University of Ottawa Press © 1951)
- Dr. Kenneth McLaughlin, Dr. Gerald Stortz, and Father Jim Wahl, 'Enthusiasm for the Truth: An Illustrated History of Saint Jerome's University', (Waterloo: University of Waterloo Press © 2002)
- Dr. Kenneth McLaughlin 'Waterloo: The Unconventional Founding of an Unconventional University' (Waterloo: University of Waterloo Press © 1997)
- Dr. Kenneth McLaughlin 'Out of the Shadow of Orthodoxy: Waterloo @ 50' (Waterloo: University of Waterloo Press © 2007)
- Flora Roy 'Recollections of Waterloo Lutheran University 1960-1973' (Waterloo: Wilfrid Laurier University Press, September 25, 2006)

External links [edit]

- Official website
- Official athletics website

Wikimedia Commons has media related to: *University of Waterloo*

V · T · E University of Waterloo [show]
V · T · E U15 universities [show]
V · T · E Post-secondary education in Ontario [show]
V · T · E Shad Valley host campuses [show]

Categories: University of Waterloo | Educational institutions established in 1957 | Association of Commonwealth Universities

This page was last modified on 23 May 2012 at 00:22.

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of use for details.
Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Contact us

Privacy policy About Wikipedia Disclaimers Mobile view

Figure 3.1: A sample Wikipedia page.

3.3.1 Titles

Each article in Wikipedia has one or more titles. Titles, as the word suggests, are the labels specifying what an article is about. There are three types of titles:

1. Main titles
2. Redirect titles
3. Disambiguation titles

To understand what each of these titles mean, each concept has been described using the example of Figure 3.2. Each article has one and only one main title and each main title only refers to one specific article. In other words, the main title is the unique key identifying an article. For instance, in the Figure 3.2, titles "United States" and "Americas" (shown by solid line boxes on the left) are the main titles pointing to their corresponding articles (shown by the large boxes on the right).

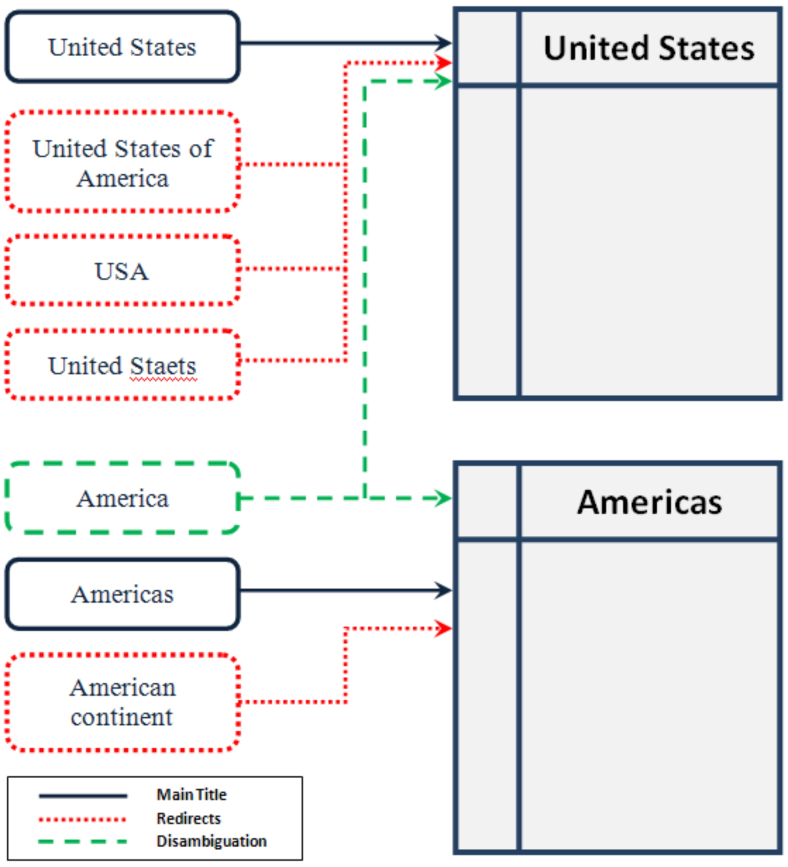


Figure 3.2: An example of the main, redirect, and disambiguation title.

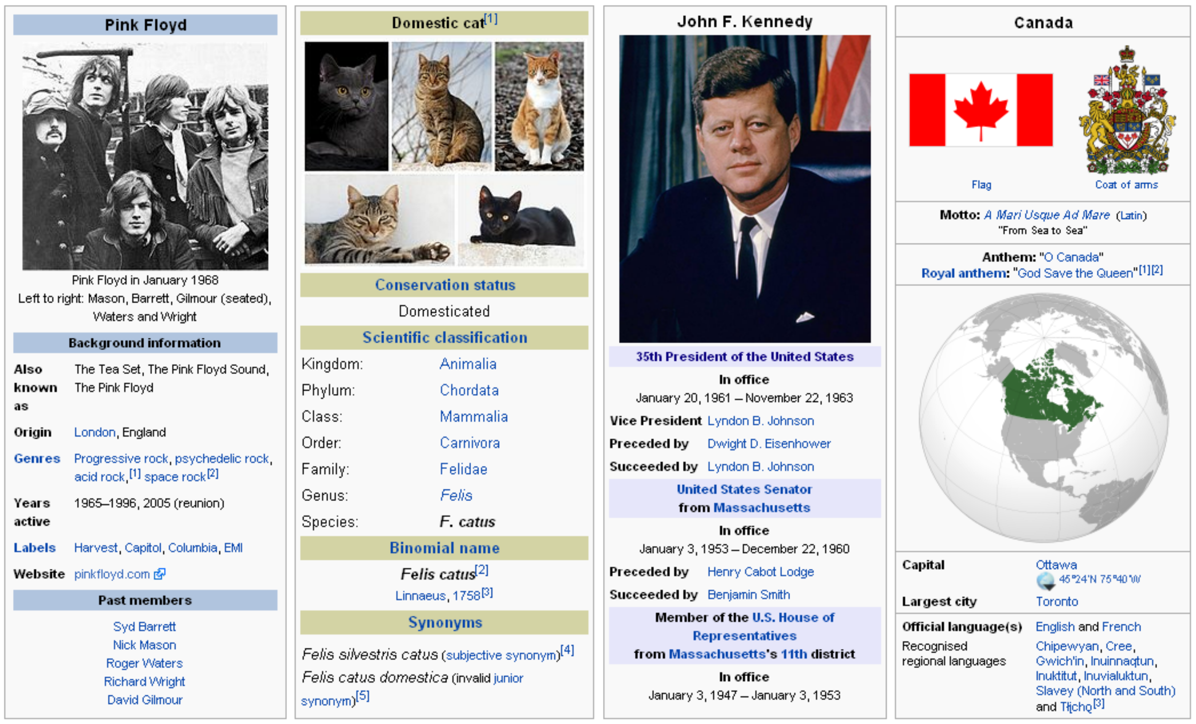
Each article might have a number of redirect titles (or simply called redirects). Each redirect points to the main title of an article and, therefore, points to the same article the main title points to. The use of redirects is to provide additional links for alternative names. The alternates can be the various names which mean the same, the foreign translations, abbreviations, and typos. In the example above, the main title “United States” has “United States of America”, “USA”, and a common typo “United Staets” as redirects. The “Americas” article, on the other hand, has a redirect called “American continent”.

The last type of titles are called disambiguation titles which are the reverse of a redirect meaning that one disambiguation title points to many different articles. Disambiguation titles are provided to differentiate the different senses of a title if the title has multiple meanings. For instance, the title “America” in Figure 3.2 might be interpreted as the United States or it might refer to the entire continent of America (North and South America). While some of the titles pointed to by disambiguation titles have different names than the disambiguation title, as shown in the above example, in many cases, there is a strong possibility that the different articles pointed to by the disambiguation title have the exact same name. For instance, “Phoenix” can refer to all of the following articles: Phoenix, a mythical bird; Phoenix, Arizona; a 1998 movie called Phoenix; and a minor constellation in the southern sky. To differentiate these types of commonality, the disambiguation title is annotated with extra tags; the annotated titles then become the main titles of the articles to which the disambiguation title was originally pointing. In the case of the “Phoenix” example, the titles will respectively be converted to the following annotated main titles: “Phoenix (mythology)”, “Phoenix, Arizona”, “Phoenix (film)”, and “Phoenix (constellation)”.

3.3.2 Infoboxes

Infoboxes are fixed-format tables which appear on some Wikipedia articles and often act as summaries of the articles. The entries provided in these boxes are often the unifying items that similar articles share e.g. capital city shared by all countries. Additionally, some of the entries are included to facilitate access and navigation to related articles. The inclusion of an infobox is neither a requirement nor banned on any given article. The decision to add the infobox or not, and if so, what kind of infobox to add and which part of it to fill is a decision made by consensus among all the collaborative authors contributing to the page.

The content of an infobox comes from certain templates which contain important facts and statistics typical to the associated articles of such templates. For instance, the articles related to people will often feature date of birth as a common field for all human beings. Some articles might also feature spouses and children if applicable. Then, for different professions, different fields are added on. Politicians have records of offices they hold and the dates they hold them while athletes' infoboxes present information about their possible awards and physical attributes. In other words, these templates work as easy-to-read synopsis of the concept presented in the article in a small ontology format. Such ontological nature of infoboxes is the main reason behind their conversion into real RDF ontology of DBpedia as we will see in Section 3.4. Figure 3.3 demonstrates four sample infoboxes using four different template types: musical bands, animals, politicians, and countries.



Pink Floyd

Pink Floyd in January 1968
Left to right: Mason, Barrett, Gilmour (seated), Waters and Wright

Background information

Also known as	The Tea Set, The Pink Floyd Sound, The Pink Floyd
Origin	London, England
Genres	Progressive rock, psychedelic rock, acid rock,[1] space rock[2]
Years active	1965–1996, 2005 (reunion)
Labels	Harvest, Capitol, Columbia, EMI
Website	pinkfloyd.com

Past members

Syd Barrett
Nick Mason
Roger Waters
Richard Wright
David Gilmour

Domestic cat[1]

Conservation status

Domesticated

Scientific classification

Kingdom:	Animalia
Phylum:	Chordata
Class:	Mammalia
Order:	Carnivora
Family:	Felidae
Genus:	*Felis*
Species:	***F. catus***

Binomial name

Felis catus[2]
Linnaeus, 1758[3]

Synonyms

Felis silvestris catus (subjective synonym)[4]
Felis catus domestica (invalid junior synonym)[5]

John F. Kennedy

35th President of the United States

In office
January 20, 1961 – November 22, 1963

Vice President	Lyndon B. Johnson
Preceded by	Dwight D. Eisenhower
Succeeded by	Lyndon B. Johnson

United States Senator from Massachusetts

In office
January 3, 1953 – December 22, 1960

Preceded by	Henry Cabot Lodge
Succeeded by	Benjamin Smith

Member of the U.S. House of Representatives from Massachusetts's 11th district

In office
January 3, 1947 – January 3, 1953

Canada

Flag
Coat of arms

Motto: *A Mari Usque Ad Mare* (Latin)
"From Sea to Sea"

Anthem: "O Canada"
Royal anthem: "God Save the Queen"[1][2]

Capital	Ottawa 45°24′N 75°40′W
Largest city	Toronto
Official language(s)	English and French
Recognised regional languages	Chipewyan, Cree, Gwich'in, Inuinnaqtun, Inuktitut, Inuvialuktun, Slavey (North and South) and Tłįchǫ[3]

Figure 3.3: Four examples of different types of infoboxes.

3.3.3 Wikilinks

Wikilinks are the hyperlink equivalents in Wiki articles. A wikilink, also called an internal link, is a link connecting a page in Wikipedia to another. It is simply formatted by double square brackets such as [[article]]. A wikilink might have some extra parameters attached. These parameters can be a wide range of editing and linking tricks for flexible mark-up purposes. Two important examples of such parameters are labels and types. Labels are often aliases describing or clarifying the link in a human readable format. For instance, "online collection of structured data" is a label for the link to the article

"Online database" mentioned in the Freebase Wikipedia article [70]. Types are tags describing what kind of page the link is pointing to. Types are specified using the type name followed by a colon and then the article or resource page name. For instance, a wikilink specified by [[file:Image_Name.png]] is a link to a page containing a digital picture file. Among these typed wikilink, the one linking to categories is of most importance to this work. Category links are described in next subsection.

3.3.4 Categories

Categories are the most important elements of an article in regards to the topic of this research. Each article is tagged by (or points to) one or more categories. The categories of Wikipedia are structurally regarded as semi-articles. Their template is different than a regular article and their links are specified using a typed wikilink as described above by "category:" tags behind their names. For instance, the category of science is defined as [[Category:Science]]. The number of categories is in the hundreds of thousands and keeps growing. In September 2008, the count was around 400,000 and by the July 2011, it grew to more than 740,000 categories.

The categories of Wikipedia are not just labels but they also have features of their own. Wikipedia considers each category as a typed article. Each category article starts with a short summary of what topic the category is representing. The body provides backward links to all the regular articles that are pointing to this category. Finally, each category can have its own category labels on the bottom of its article page similar to the categories of a regular article demonstrated by label 4 in Figure 3.1. This type of structure will cause each category to belong to another parent category and, therefore, form a hyponymy graph.

Wikipedia's categories describe every domain of knowledge ranging from the very precise, such as "fictional secret agents and spies", to the very general, such as "information". As explained, the categories are connected by hyponym relationships, with a child category having an "is-a" relationship to its parents. Therefore, more specific categories have more general parents called super-categories. However, the graph is not strictly hierarchic: there exist shortcuts in the connections (i.e. starting from one child category and going up two different paths of different lengths to reach the same parent category) as well as loops (i.e. starting from one child category and going up a path to reach the same child category again). Some of these characteristics, such as shortcuts, are tolerable but some, like loops, present serious issues in the semantic analysis and, therefore, need some pre-processing before they can be used. Section 3.5 will explore the details of the category graph and provide information on how the graph is pruned and processed to be used in analysis.

3.3.5 Other Elements

A Wikipedia article has a number of other elements which are not significantly used in this research. Before moving on, it is worth briefly describing some of these elements and pointing out some useful applications which can be performed using these elements. These elements include, but are not limited to:

- Summary
- Table of Contents
- Body
- “See Also” section
- References & External links
- Templates

The body of an article is the most important element of the article. While the wikilinks are embedded in different parts of the article body, the statistical analysis of the body words can be a useful resource. For instance, calculating the frequency of each of the words in an article and relating those frequencies to the categories of the article can act as a potential approach for document classification problems.

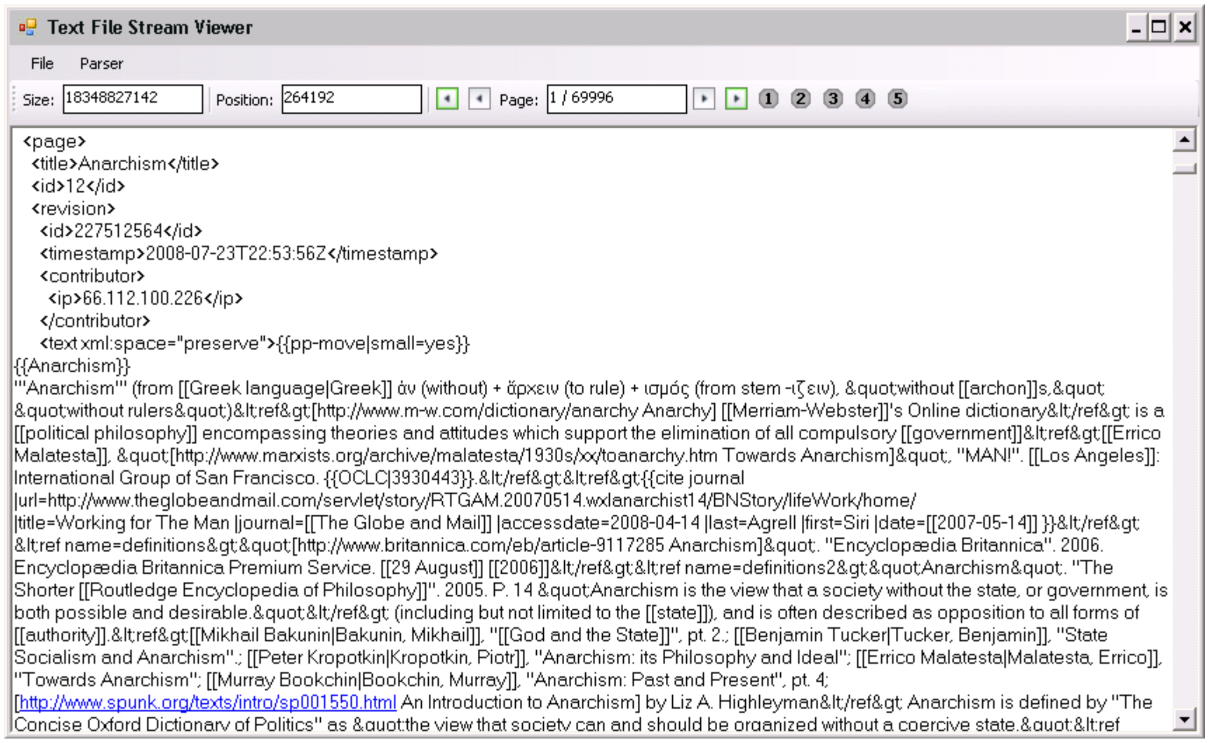
The Table of Contents (ToC) can be used to group wikilinks and annotate them with semantic tags. Specifically, the headers of each section are useful clues to tag wikilinks. The “See Also” section is part of the ToC which usually groups wikilinks of certain articles which are closely related to the current article. The next subsection will explore a little further about how the ToC can be used to extract RDF triples.

The summary, especially its first sentence, is a good indicator of the main concept that the article subject is related to. For instance, Albert Einstein is affiliated with many topics based on the categories of his Wikipedia article such as Nobel laureate, inventor, and humanitarian. But the first sentence of the summary of his article specifically introduces him as “a German theoretical physicist” which is the first title people mostly associate him to.

Lastly, the templates are common structures, possibly following a timeline, that many related articles are associated with. For instance, the family template indicates the spouse(s) and child(ren) of a person and can be used to generate important RDF triples automatically and with high accuracy.

3.3.6 Challenges

Although Wikipedia looks like a very regularly structured data source compared to common hyper-text web pages, there are still a few challenges to be dealt with in order to efficiently use Wikipedia for automated semantic analysis purposes. The main problem regarding Wikipedia is that, despite all the structures, templates, tags, and mark-ups, Wikipedia is still a resource designed primarily for human-readability not machine-readability. The presence of all the elements mentioned above is, for the most part, intended for enhancement of usability and navigation. These elements are user interface features that help human users to find relevant information faster and easier. They are not, on the other hand, engineered to be a computing resource by nature and using them in a computing task requires further processing.



Size: 18348827142 Position: 264192 Page: 1 / 69996

```
 <page>
  <title>Anarchism</title>
  <id>12</id>
  <revision>
   <id>227512564</id>
   <timestamp>2008-07-23T22:53:56Z</timestamp>
   <contributor>
    <ip>66.112.100.226</ip>
   </contributor>
   <text xml:space="preserve">{{pp-move|small=yes}}
{{Anarchism}}
'''Anarchism''' (from [[Greek language|Greek]] ἀν (without) + ἄρχειν (to rule) + ισμός (from stem -ιζειν), &quot;without [[archon]]s,&quot;
&quot;without rulers&quot;)&lt;ref&gt;[http://www.m-w.com/dictionary/anarchy Anarchy] [[Merriam-Webster]]'s Online dictionary&lt;/ref&gt; is a
[[political philosophy]] encompassing theories and attitudes which support the elimination of all compulsory [[government]]&lt;ref&gt;[[Errico
Malatesta]], &quot;[http://www.marxists.org/archive/malatesta/1930s/xx/toanarchy.htm Towards Anarchism]&quot;, "MAN!". [[Los Angeles]]:
International Group of San Francisco. {{OCLC|3930443}}.&lt;/ref&gt;&lt;ref&gt;{{cite journal
|url=http://www.theglobeandmail.com/servlet/story/RTGAM.20070514.wxlanarchist14/BNStory/lifeWork/home/
|title=Working for The Man |journal=[[The Globe and Mail]] |accessdate=2008-04-14 |last=Agrell |first=Siri |date=[[2007-05-14]] }}&lt;/ref&gt;
&lt;ref name=definitions&gt;&quot;[http://www.britannica.com/eb/article-9117285 Anarchism]&quot;. "Encyclopædia Britannica". 2006.
Encyclopædia Britannica Premium Service. [[29 August]] [[2006]]&lt;/ref&gt;&lt;ref name=definitions2&gt;&quot;Anarchism&quot;. "The
Shorter [[Routledge Encyclopedia of Philosophy]]". 2005. P. 14 &quot;Anarchism is the view that a society without the state, or government, is
both possible and desirable.&quot;&lt;/ref&gt; (including but not limited to the [[state]]), and is often described as opposition to all forms of
[[authority]].&lt;ref&gt;[[Mikhail Bakunin|Bakunin, Mikhail]], "[[God and the State]]", pt. 2.; [[Benjamin Tucker|Tucker, Benjamin]], "State
Socialism and Anarchism".; [[Peter Kropotkin|Kropotkin, Piotr]], "Anarchism: its Philosophy and Ideal"; [[Errico Malatesta|Malatesta, Errico]],
"Towards Anarchism"; [[Murray Bookchin|Bookchin, Murray]], "Anarchism: Past and Present", pt. 4;
[http://www.spunk.org/texts/intro/sp001550.html An Introduction to Anarchism] by Liz A. Highleyman&lt;/ref&gt; Anarchism is defined by "The
Concise Oxford Dictionary of Politics" as &quot;the view that society can and should be organized without a coercive state.&quot;&lt;ref
```

Figure 3.4: The developed Wikipedia parser tool

The first issue is that of parsing the data. Although Wikipedia dump files [71] provide SQL statements so that the end user can regenerate a database, the provided links are too untidy to use. For example, there are a large number of articles and categories that are created and kept for editing and maintenance purposes. These data do not contribute to the main encyclopedic purpose of Wikipedia and one needs to write some parsing code to clean the data from these extra materials.

Alternatively, one might use the textual format of the Wikipedia database dump which is free of these extra entries. This alternate format is one giant XML file that carries each article as a XML tagged element. Therefore, aside from a few main pieces of information, such as the title of an article, the remainder of each article body should be parsed so that the information from all the wikilinks, categories, and infoboxes gets extracted. In any case, some effort should be put into converting Wikipedia resources to machine-friendly versions. Figure 3.4 shows a parser tool developed in early stages of this research work by the author for this purpose.

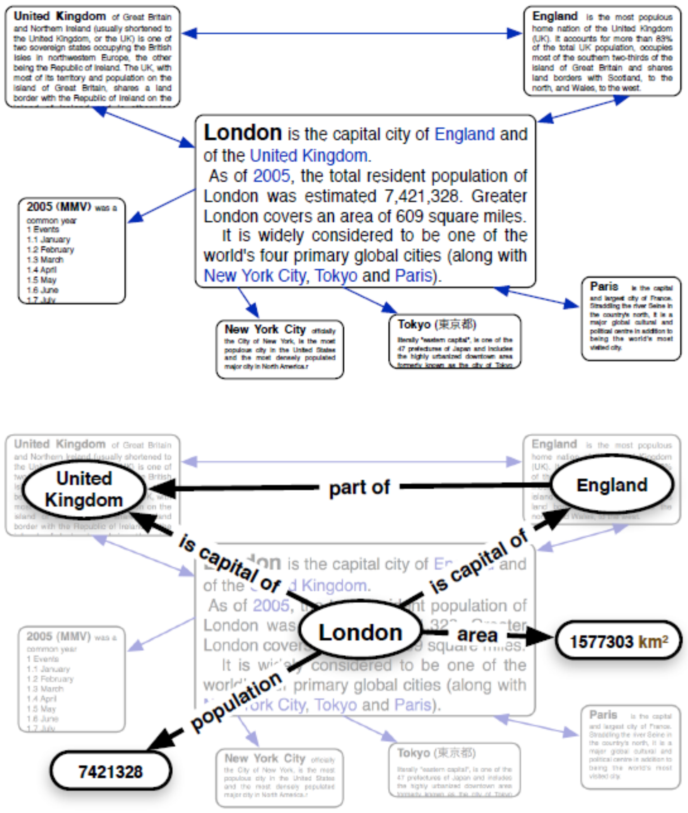

United Kingdom
England
London is the capital city of England and of the United Kingdom.
As of 2005, the total resident population of London was estimated 7,421,328. Greater London covers an area of 609 square miles.
It is widely considered to be one of the world's four primary global cities (along with New York City, Tokyo and Paris).
2005 (MMV)
New York City
Tokyo (東京都)
Paris
part of
is capital of
area
population
London
1577303 km²
7421328

Figure 3.5: Converting wikilinks to RDF triples (adopted from [50]).

Assuming that the basic elements of Wikipedia, such as article titles, wikilinks, and categories, are successfully extracted, the data is not yet fully proper for semantic uses. The lack of proper predicate labels for wikilinks is one of the major semantic issues. Wikilinks provide links between two articles. If we consider articles as entities, a wikilink simply asserts that the source article is related to the target article. The problem is that the type of these relations is not specified. As an example, the Wikipedia article about Toyota points to Japan but we cannot derive what the relationship between Toyota and Japan is solely based on this link. To compensate this deficiency, many researchers tried to encourage the use of semantic Wiki framework so that the authors can easily integrate the type of the links as well. Works of [50, 72] along with the WikSAR project [73], Platypus Wiki [74, 75], and Rhizome [76] all more or less suggest converting wikilinks to RDF triples by mapping the article title to the subject, the linked article to the object, and the type of the link to the predicate part of the RDF triple. Obviously, the type should be provided by the author beforehand. Figure 3.5 shows an example of converting wikilinks to RDFs.

However, there are intuitive approaches which can try to detect or estimate the type of the wikilinks automatically. As mentioned earlier, the Table of Contents can be one resource. The wikilinks are distributed in the article body and if extracted normally, the order and position of the wikilinks would be lost. But if, during parsing the body, the header of the section in which the link appears is recorded, it is at least possible to have a general idea of what that link is about. For instance, the link to "Ulm" in the Einstein article appears in the "Early life and education" section which is not far from the exact type being his birth place. [77] provides other uses of the ToC for partitioning the text in order to find the most relevant patterns. Although grouping and tagging wikilinks with their ToC section header gives a general idea of the topic of the wikilink, a more sophisticated approach, such as natural language processing, should be used along with the ToC to detect the type of a wikilink. One simple method is to find the closest verb or noun of the wikilink and use that as an estimate. For instance, in sentences: "Albert Einstein was born in *Ulm*" and "His father was *Hermann Einstein*", the verb 'born' and the noun 'father' ('was' is ignored as being a form of 'to be': a stop word), are closest words which define two wikilinks, *Ulm* and *Hermann Einstein*.

Finally, the very important step to be taken to improve the semantic value of the knowledgebase is the processing of the infobox information. If processed accurately, infobox information can provide immense semantic merit to the knowledgebase. Processing the infobox is a specifically challenging task. Below are a few of the issues to be noted when parsing infoboxes to efficiently extract the data:

- There are many different infobox templates.
- In each template, some fields might be missing or not applicable. For instance, some people might be alive and some might not and, therefore, a date of death might not be included in some articles.
- Some of the entries in infoboxes are pointers to other articles while some others are literal values.

Despite the above difficulties, the tabular nature of infoboxes has invoked interest in a number of research works. Auer and Lehmann [78] show that forming semantic RDF triples by converting infoboxes can be done easily and then be used in astounding question-answering tasks. Wu and Weld, on the other hand, introduce Kylin [79], a self supervised machine learning system that extracts semantic relationships from Wikipedia from natural language text, constructs and completes infoboxes, and identifies missing links for proper nouns on each page. Later, they introduced the KOG autonomous system [80] to further refine infoboxes in order to achieve a clean structure ontology. Moreover, in the *Intelligence in Wikipedia* project [81], they try to extract a knowledgebase of semantic triples by combining two paradigms called information extraction (IE) and communal content creation (CCC) which are again based on infobox information. Last, but certainly not least, the DBpedia project [68, 82] is a very famous example of successfully converting infobox information into RDF triples. The next section will give a meticulous description of the DBpedia structure and its components.

3.4 DBpedia

Simply said, DBpedia is the semantic version of Wikipedia; or at least is one possible adaptation of it. Although it is not the answer to all of Wikipedia's problems mentioned above and although it does not have every desirable semantic entry, such as strongly typed RDF triples of wikilinks mentioned earlier, it provides the best machine-readable Semantic Web experience out of Wikipedia so far. DBpedia is engineered to be used in computing tasks, specifically in semantic analysis methods; no wonder it has become the *nucleus* [68] and *crystallization point* [82] for the Web of Data (Linked Data) initiative. In this section, some of the more important DBpedia components are discussed. Many of the concepts are, however, based on Wikipedia's counterparts that are already explained in the last section. Therefore, this section delves a bit more into formatting of the data rather than the definition of the concepts.

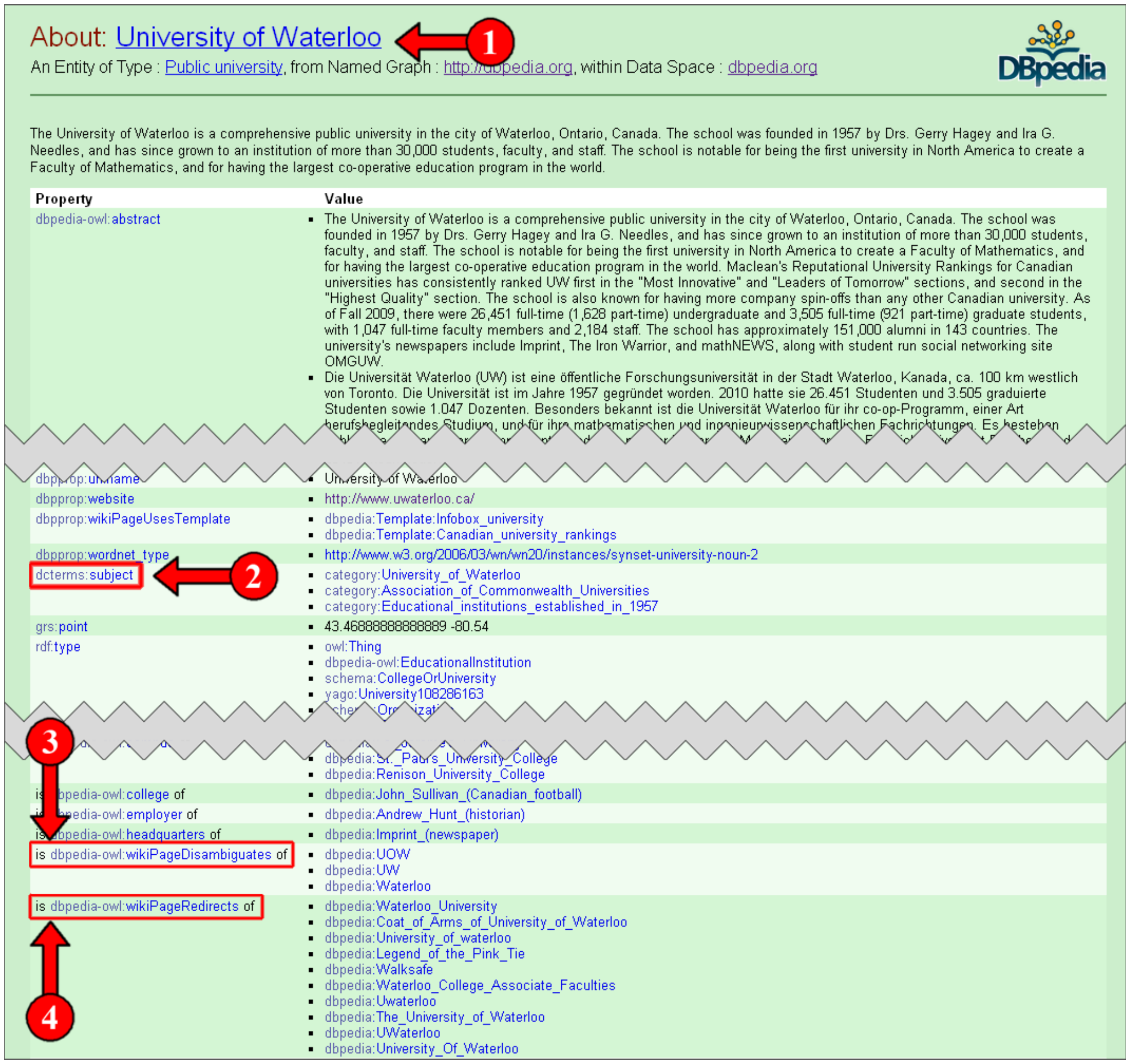
The DBpedia community's goal is to extract structured information from Wikipedia and make it available on the web for sophisticated queries [7]. Therefore, DBpedia is trying to address challenges mentioned in the previous section. In fact, it does more than that. Not only does it clean up the garbage information, like maintenance articles, and provides intelligently designed RDF triples for different elements of Wikipedia, it also annotates the extracted information by linking them to other knowledgebases like YAGO [46, 47], hence creating the linked data.

DBpedia provides multiple features and resources, namely: the knowledge ontology, external links to many other data sources, a SPARQL [38] query portal, and natural language processing datasets. However, the main core of DBpedia is the collection of semantic datasets. These datasets represent various elements of Wikipedia, some of which were mentioned in the previous section, and the relations between them. Time-wise, they are extracted from a snapshot dump of the Wikipedia database [71] at a certain date but they are continually getting updated in a batch operation once the new versions of Wikipedia dumps become available. Recently, DBpedia launched DBpedia Live [83] to make its resources even more up-to-date. DBpedia Live keeps updating the datasets as soon as new Wikipedia articles become available instead of waiting for an entire new dump to be released.

The structure of these datasets is, in fact, what makes them interesting for computing. They are sets of triples in each line adherent to the popular RDF structure of *subject-predicate-object*. There are also quadruple versions of the dataset available with an additional fourth column specifying the source of the triple. The complete list of these dataset and their description are available at [84, 85]. Here, we take a look at the important ones which are also used in this methodology:

- Titles
- Redirects
- Disambiguation Links
- Ontology Infobox Types
- Ontology Infobox Properties
- Wikipedia Pagelinks
- Categories (SKOS)
- Article Categories

Before describing each dataset individually, it is noteworthy to take a look at a big picture showing how different entries of different datasets make up a DBpedia page equivalent of the Wikipedia article. Figure 3.6 portrays a sample DBpedia page which corresponds to the Wikipedia article of Figure 3.1 about the University of Waterloo. The Wikipedia article is located at `<http://en.wikipedia.org/wiki/University_of_Waterloo>` while the DBpedia page is located at `<http://dbpedia.org/page/University_of_Waterloo>`. We can see that the format of the DBpedia URL is based on the Wikipedia one and a similar rule is applied in other languages.



About: University of Waterloo

An Entity of Type : Public university, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

The University of Waterloo is a comprehensive public university in the city of Waterloo, Ontario, Canada. The school was founded in 1957 by Drs. Gerry Hagey and Ira G. Needles, and has since grown to an institution of more than 30,000 students, faculty, and staff. The school is notable for being the first university in North America to create a Faculty of Mathematics, and for having the largest co-operative education program in the world.

Property	Value
dbpedia-owl:abstract	▪ The University of Waterloo is a comprehensive public university in the city of Waterloo, Ontario, Canada. The school was founded in 1957 by Drs. Gerry Hagey and Ira G. Needles, and has since grown to an institution of more than 30,000 students, faculty, and staff. The school is notable for being the first university in North America to create a Faculty of Mathematics, and for having the largest co-operative education program in the world. Maclean's Reputational University Rankings for Canadian universities has consistently ranked UW first in the "Most Innovative" and "Leaders of Tomorrow" sections, and second in the "Highest Quality" section. The school is also known for having more company spin-offs than any other Canadian university. As of Fall 2009, there were 26,451 full-time (1,628 part-time) undergraduate and 3,505 full-time (921 part-time) graduate students, with 1,047 full-time faculty members and 2,184 staff. The school has approximately 151,000 alumni in 143 countries. The university's newspapers include Imprint, The Iron Warrior, and mathNEWS, along with student run social networking site OMGUW. ▪ Die Universität Waterloo (UW) ist eine öffentliche Forschungsuniversität in der Stadt Waterloo, Kanada, ca. 100 km westlich von Toronto. Die Universität ist im Jahre 1957 gegründet worden. 2010 hatte sie 26.451 Studenten und 3.505 graduierte Studenten sowie 1.047 Dozenten. Besonders bekannt ist die Universität Waterloo für ihr co-op-Programm, einer Art berufsbegleitendes Studium, und für ihre mathematischen und ingenieurwissenschaftlichen Fachrichtungen. Es bestehen [illegible]
dbpprop:uniname	▪ University of Waterloo
dbpprop:website	▪ http://www.uwaterloo.ca/
dbpprop:wikiPageUsesTemplate	▪ dbpedia:Template:Infobox_university ▪ dbpedia:Template:Canadian_university_rankings
dbpprop:wordnet_type	▪ http://www.w3.org/2006/03/wn/wn20/instances/synset-university-noun-2
dcterms:subject	▪ category:University_of_Waterloo ▪ category:Association_of_Commonwealth_Universities ▪ category:Educational_institutions_established_in_1957
grs:point	▪ 43.4688888888889 -80.54
rdf:type	▪ owl:Thing ▪ dbpedia-owl:EducationalInstitution ▪ schema:CollegeOrUniversity ▪ yago:University108286163 ▪ schema:Organization
[illegible]	▪ [illegible] ▪ dbpedia:St._Paul's_University_College ▪ dbpedia:Renison_University_College
is dbpedia-owl:college of	▪ dbpedia:John_Sullivan_(Canadian_football)
is dbpedia-owl:employer of	▪ dbpedia:Andrew_Hunt_(historian)
is dbpedia-owl:headquarters of	▪ dbpedia:Imprint_(newspaper)
is dbpedia-owl:wikiPageDisambiguates of	▪ dbpedia:UOW ▪ dbpedia:UW ▪ dbpedia:Waterloo
is dbpedia-owl:wikiPageRedirects of	▪ dbpedia:Waterloo_University ▪ dbpedia:Coat_of_Arms_of_University_of_Waterloo ▪ dbpedia:University_of_waterloo ▪ dbpedia:Legend_of_the_Pink_Tie ▪ dbpedia:Walksafe ▪ dbpedia:Waterloo_College_Associate_Faculties ▪ dbpedia:Uwaterloo ▪ dbpedia:The_University_of_Waterloo ▪ dbpedia:UWaterloo ▪ dbpedia:University_Of_Waterloo

Figure 3.6: A sample DBpedia page.

The DBpedia URL is, however, not the main unique key to identify its resources. It just provides a human-readable version of its resources for each article. Each URL has a standard resource URI pointing to it which is the main identifier of any given DBpedia resource. We saw that RDF format is fundamentally based on URIs. In DBpedia each resource is identified by a URI formatted as: `<http://dbpedia.org/resource/unique_resource_name>`. The rationale is that URIs act as the identifiers of the resources and then, if any URI is entered into a browser, it will redirect to a URL so that the user lands on a webpage with human-readable material describing the resource. This method, in fact, is the standard RDF design in the Semantic Web community [18].

Consequent to above discussion, the page shown in Figure 3.6 is referring to a unique resource. The resource URI of this page is then the identifier occurring in all the datasets. Since all the information on this page are characteristics related to the presented resource, it stands to reason that the resource would be the subject of any RDF triple presented in this page. This assumption is true because the page is actually created based on all the triples which have the given resource URI as their subject. The page generation, in effect, gathers all the triples with the resource URI as subject and creates a two columned table with first column being the predicate and the second column being the object. The numerical labels on the figure point to some of the fields of datasets which are explained below. The title, labeled as 1, corresponds to the same label in Figure 3.1. The entries in Figure 3.6 exist in different datasets. In the following subsections, the aforementioned main datasets are described and marked where the labeled entries in the figure are located.

3.4.1 Titles Dataset

The titles dataset, as the name suggests, is the dataset that presents all the titles and their URIs. None of the typed or maintenance titles are included here and all the presented titles are the ones related to knowledge items. This dataset is, therefore, used as the basis of our entity graph as explained in Section 3.5 and it constitutes one of the pillars of the semantic analysis methodology. The format of the titles dataset is simply designed to connect the URI of each title to its English language label. The triple is as follows[1]:

```
<Title_URI> <http://www.w3.org/2000/01/rdf-schema#label> "Title Label"@en .
```

`<Title_URI>` is the title identifier through all other datasets. The middle URI is the W3C standard predicate representing the concept of “label”. Lastly, the “Title Label” is a literal string value

[1] Subject and object URIs are shown in blue, predicate URIs in red, and literals in green.

indicating the English label of the title. The @en notion indicates the English language. Each row is terminated with a period. The triple is indicating, by the RDF format, that the entity known as `<Title_URI>` has a label called “Title Label”. An example for “Albert Einstein” is below:

```
<http://dbpedia.org/resource/Albert_Einstein> <http://www.w3.org/2000/01/rdf-
schema#label> "Albert Einstein"@en .
```

Each line in the titles dataset is associated with a page such as that in Figure 3.6. The URI is mapped to the URL of the page and the label is displayed at the location marked 1 on the figure.

3.4.2 Redirects Dataset

The redirects dataset represent a subset of the titles dataset and indicates which of the titles are redirect ones. It also provides the main title each redirect title points to. The format is:

```
<Redirect_Title_URI> <http://dbpedia.org/ontology/wikiPageRedirects>
<Main_Title_URI> .
```

The predicate URI has been created by DBpedia (along with many others in its ontology). The redirect title URI and main title URI each have one corresponding row in the titles dataset. Each redirect title points to only one main title and as a result has only one row in this dataset. An example for a redirect title can be:

```
<http://dbpedia.org/resource/Waterloo_University> <http://dbpedia.org/ontology/
wikiPageRedirects> <http://dbpedia.org/resource/University_of_Waterloo> .
```

Figure 3.6 shows the instances of redirect entries marked as 4 with the first row corresponding to the above example. The URI on the left column is the predicate URI which was shown above and the ones on the right column are different redirect titles. This means that the order is reversed in the figure; this is done so that the main title is the subject of all triples featured on the page, as discussed earlier. Therefore, to show this swapped direction, an “of” is added after the predicate URI.

3.4.3 Disambiguation Links Dataset

The disambiguation links dataset is very similar to the redirect dataset and indicates the disambiguation titles and the various main titles that the disambiguation title clarifies. The format is:

```
<Disambiguation_Title_URI> <http://dbpedia.org/ontology/wikiPageDisambiguates>
<Main_Title_URI> .
```

Figure 3.6 shows some inverted examples marked 3. The only difference with the redirects dataset is the cardinality of the relations. Each disambiguation title naturally disambiguates many main titles while each redirect title redirects to only one main title. For instance, the title “UW”, can point to both “University of Waterloo” and “University of Washington”.

3.4.4 Ontology Infobox Types Dataset

This dataset maps titles to a list of ontological types extracted from infoboxes. The list features 320 different *classes* of things at the time of writing [86]. These types can be anything from professions such as Artists, Athletes, and Governors to region types such as City, Town, and Country. The dataset not only maps the titles to one of those 320 types/classes but it might provide equivalent triples mapping the titles to the types of other sources such as YAGO [47, 82]. The format is:

```
<Title_URI> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <Type_URI> .
```

Types are similar to categories but they are very general and tend to feature the main characteristics of the entity they are presenting. “University of Waterloo” is of type *University* while “Aristotle” is both a *Person* and a *Philosopher*.

3.4.5 Ontology Infobox Properties Dataset

As discussed in last section, infoboxes are valuable sources of structured information in some articles. This dataset is the result of careful conversion of article infoboxes into RDF triples. As mentioned before, infoboxes provide values for certain common characteristics of an entity, e.g. birth date of people. Considering the article as an entity, these characteristics are called properties of the entity. Their values can either be literal values or other entities. Since the RDF object can be either a literal or a URI, such structure fits nicely into a RDF triple format as below:

```
<Title_URI> <Property_URI> “Value”[^^Datatype_URI][@en] .
```

```
<Title_URI> <Property_URI> <Value_Title_URI> .
```

The `<Title_URI>` refers to the main title (entity) and the `<Property_URI>` specifies what property (characteristic) of the title is presented. If the value of the property is a literal, it will be expressed in quotations; otherwise, if the value is an entity, its URI is given. In case of a literal value, it is possible that the data type of the value is followed and if the value is a string, the language (here English) will be added. There is a special case of homepages where the value URI will be replaced by the URL of the homepage as the following depicts:

```
<Title_URI> <http://xmlns.com/foaf/0.1/homepage> <URL> .
```

The interesting notion about the properties dataset is that the predicate URI varies so that it can express different properties. DBpedia, as of this moment, provides more than 1600 different property URIs in its ontology and also borrows some property URIs, such as the homepage above, from other RDF sources. The following are a few examples of property triples expressing Canada's leader title, founding date, and currency:

```
<http://dbpedia.org/resource/Canada> <http://dbpedia.org/ontology/leaderTitle>
“Prime Minister”@en .

<http://dbpedia.org/resource/Canada> <http://dbpedia.org/ontology/foundingDate>
"1867-07-01"^^xsd:date .

<http://dbpedia.org/resource/Canada> <http://dbpedia.org/ontology/currency>
<http://dbpedia.org/resource/Canadian_dollar> .
```

3.4.6 Wikipedia Pagelinks Dataset

The pagelinks dataset contains the wikilinks. The format is:

```
<Source_Title_URI> <http://dbpedia.org/ontology/wikiPageWikiLink>
<Target_Title_URI> .
```

The number of wikilinks, as of the time of writing, is about 146 million triples. The links are directed, meaning, for example, “University of Waterloo” points to “Research In Motion” but not the other way around. However, it is possible that both titles point to each other e.g. “University of Waterloo” points to “Waterloo, Ontario” and vice versa. This dataset is a rich source for data mining and ripe for algorithms similar to Page Rank.

3.4.7 Categories (SKOS) Dataset

The categories dataset is, in fact, the most important dataset used in this work. It counts for the topic graph portion of the semantic graph presented in Section 3.5. This dataset provides the list of Wikipedia's categories (about 740,000) along with their hyponym relations. The dataset uses Simple Knowledge Organization System (SKOS) vocabulary [87] to describe categories and their relations. A complete list of SKOS vocabulary and its ontology is given in its W3C recommendation page [88]. However, this dataset only uses three classes of SKOS ontology: skos:preflabel, skos:Concept, and skos:broader[2]. These three are used to describe three definition parts which each category in this dataset has. For each category, one row provides the human-readable label (skos:preflabel) of the category (similar to the titles dataset labels but using SKOS vocabulary). The second part is another row common for all categories which indicates that each category is a concept (skos:Concept). This row is to maintain the RDF well-formedness and semantic correctness. The third part is a set of rows showing all the super-categories of the given category with broader semantics (skos:broader) which in turn produces the hyponymy graph. These three parts have the following formats:

```
1. <Category_URI> <http://www.w3.org/2004/02/skos/core#prefLabel>
   "Category_Label"@en .

2. <Category_URI> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.w3.org/2004/02/skos/core#Concept> .

3. <Category_URI> <http://www.w3.org/2004/02/skos/core#broader>
   <Super_Category_URI> .
```

To find the sub-categories of a certain category, one should match the category URI to the object part of the skos:broader entries. The following examples show the definition of the category of "Software Engineering" and one of its super-categories, "Computer Science", and one of its sub-categories, "Computer Programming", in this dataset:

```
<http://dbpedia.org/resource/Category:Software_engineering>
<http://www.w3.org/2004/02/skos/core#prefLabel> "Software engineering"@en .

<http://dbpedia.org/resource/Category:Software_engineering> <http://www.w3.org/
1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2004/02/skos/core#Concept> .

<http://dbpedia.org/resource/Category:Software_engineering>
<http://www.w3.org/2004/02/skos/core#broader>
<http://dbpedia.org/resource/Category:Computer_science> .

<http://dbpedia.org/resource/Category:Computer_programming>
<http://www.w3.org/2004/02/skos/core#broader>
<http://dbpedia.org/resource/Category:Software_engineering> .
```

[2] The "skos:" namespace is equal to the <http://www.w3.org/2004/02/skos/core#> prefix.

3.4.8 Articles Categories Dataset

The articles-categories dataset is the part that completes the semantics graph by connecting the articles main titles to their categories. Section 3.5 will present an in-depth description of the relations created by the dataset and the pieces making up the semantic graph. The format of the dataset is:

```
<Title_URI> <http://purl.org/dc/terms/subject> <Category_URI> .
```

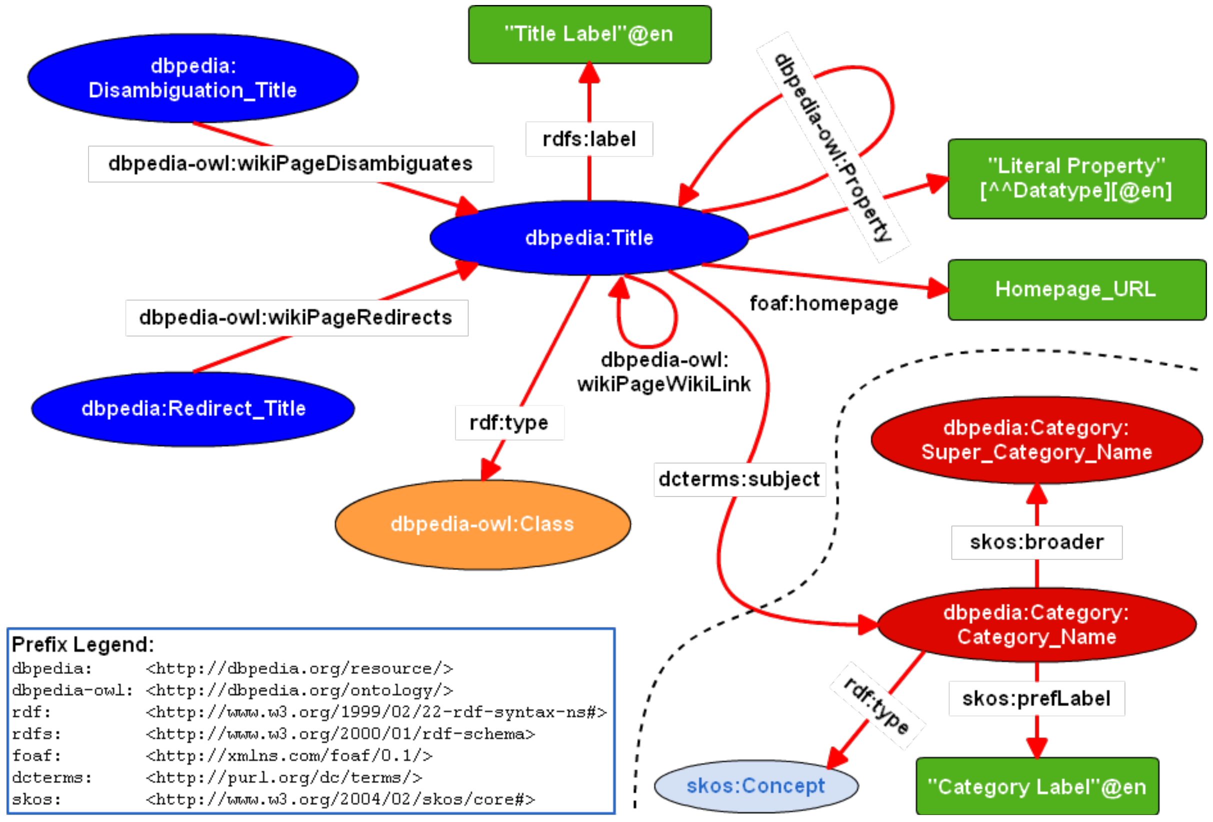
Figure 3.6 shows some examples of these triples, marked as 2. Below is another example, in triple format, showing that “Albert Einstein” belongs to a category called “Cosmologists”:

```
<http://dbpedia.org/resource/Albert_Einstein>
<http://purl.org/dc/terms/subject>
<http://dbpedia.org/resource/Category:Cosmologists> .
```

3.4.9 Final Knowledgebase

Having gathered all the DBpedia datasets mentioned above, we have enough data in our arsenal to create the appropriate knowledgebase for the semantic analysis. Note that there are many other datasets offered by DBpedia, such as Short Abstracts and Geographic Coordinates, which have been left out due to lack of use in the presented semantic analysis. However, these extra datasets can provide valuable aid in certain applications which are outside the scope of this work. Moreover, note that the quantity of the data, namely the number of triples in each dataset, keeps growing as DBpedia gets more updates from newer Wikipedia dumps. While this is a desirable feature, the additional data might not necessarily improve the performance of semantic analysis in every case. We will observe in Chapter 4 that the knowledgebase used in some experimental results is of earlier versions of Wikipedia. The only time constraint is that the test set used in those experiments predates the version of Wikipedia to avoid featuring entities which have not been yet incorporated in Wikipedia.

Figure 3.7 shows a diagram of how all the above components are linked together to create the knowledgebase. The blue ovals, green boxes, and red ovals show titles/entities, literal values, and categories/topics, respectively. Types/classes are shown by an orange oval and the skos:Concept, a constant for all categories, by a light blue oval. The namespaces used to shorten the URIs are given in the box at the bottom left corner. Notice how the dashed line separates the knowledgebase into two areas: the top left corner area with titles at its core (excluding the orange oval of classes) represents the entity graph while the bottom right corner area with categories at its core represents the topic graph. These two together constitute the semantic graph explained in Section 3.5.



Prefix Legend:

dbpedia:	<http://dbpedia.org/resource/>
dbpedia-owl:	<http://dbpedia.org/ontology/>
rdf:	<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
rdfs:	<http://www.w3.org/2000/01/rdf-schema>
foaf:	<http://xmlns.com/foaf/0.1/>
dcterms:	<http://purl.org/dc/terms/>
skos:	<http://www.w3.org/2004/02/skos/core#>

Figure 3.7: Final knowledgebase generated by DBpedia datasets.

Figure 3.7 shows the structure of the knowledgebase. However, to have a quantitative understanding of the knowledgebase volume, it is valuable to look at some statistics given in Table 3.2. These statistics belong to the DBpedia version 3.7 datasets.

Table 3.2: DBpedia 3.7 datasets statistics [86, 89].

Dataset	**Triples**	**Distinct**
Titles	8.8 million	3.6 million Main Titles
Redirects	5.1 million	
Disambiguation Links	1.0 million	~0.1 million Disambiguation Titles
Ontology Infobox Types	9.3 million	320 Types for 1.8 million of Main Titles
Ontology Infobox Properties	17.5 million	1643 Properties + ~10 External Properties
Wikipedia Pagelinks	145.9 million	
Categories (SKOS)	3.0 million	740,000 Categories
Article Categories	13.6 million	

To understand how the above statistics play into the knowledgebase, there are a few points to discuss. Firstly, note that there are 8.8 million titles but only 3.6 million of them are unique entities. From the remaining 5.2 million titles, 5.1 million of them are redirects and about 0.1 million of them are disambiguation titles. This means this knowledgebase has information about 3.6 million unique things which we will call entities. However, there are 5.2 million other access routes to these entities. These alternate routes are especially helpful in the entity detection task described in Section 3.6. Also note that the total number of triples in disambiguation links dataset is 1.0 million which is because each of those 0.1 unique disambiguation titles point to many different main titles and hence more of the total triples.

Secondly, there are 9.3 and 17.5 million triples connecting main titles to 320 different types and 1643 properties, respectively. However, the infoboxes are not available for all the main titles and, as the table indicates, these triples are associated with only 1.8 million of the main titles. This means each main title with an infobox has 5.2 types and 9.7 properties on average.

Lastly, the wikilinks connections create a graph with about 146 million edges. 5.1 million triples in the pagelinks dataset are repeats of redirects dataset rows. Therefore, excluding redirect wikilinks, the pagelinks graph has about 141 million edges for 3.6 million main titles as nodes. On the other side, the category graph has 3.0 million triples. As explained, each category has two rows specifying the label and skos:Concept connections and the rest are hyponym relations. Thus, the number of edges in the category graph is 3.0 million minus 740,000 categories twice which leaves about 1.5 million edges for 740K categories as nodes. Finally, the connection between title and category graphs is created by 13.6 million links from the titles dataset to the categories dataset. Since there are 3.6 million main titles and 740,000 categories, on average, each title has 3.8 categories and each category is related to 18.4 titles.

The above statistics conclude the description of the knowledgebase. The next section will talk about the semantic graph as a subset of the knowledgebase. It then officially introduces the definitions of entity and topic and describes the graphs associated with each one. Once the graphs are explained, some of the challenges regarding the use of the semantic graph in the methodology are presented and discussed.

3.5 Semantic Graph

The knowledgebase created from DBpedia provides a collection of entries of a diverse nature connected to one another to form a graph. The edges of the resulting graph represent semantic connections between various entries, hence producing a semantic graph. Such a semantic graph, in turn, can be used in semantic processes. However, because of the diversity of the types of the graph nodes, any analysis of the graph should be performed separately on sub-parts of the semantic graph. This section introduces and specifies two main parts of the semantic graph: entity graph and topic graph.

The basic blocks of an encyclopedia such as Wikipedia are the articles presented in it. These articles provide information about certain concepts which we call *entities*. Entities are atomic elements indicating objects, people, organizations, location, events, acts, situations, expressions, abstract concepts, laws, and so on. Any specific element can be thought of as an entity. In other words, each entity is a knowledge item and the collection of entities is assumed to constitute the human knowledge. The range and specificity of items covered by entities will differ from one knowledgebase to another. For instance, a medical knowledgebase might cover different diseases, medicines, and medical procedures extensively while not presenting any insight into any other subjects. Wikipedia is one of the largest generic knowledgebases available and, although it might not provide as many entries about different diseases as a medical knowledgebase would, it provides enough entities on any given subject to be useful in general purpose applications.

While the entities define the extent of the knowledgebase, they do not contribute to the building of the structure of the knowledgebase. A well-designed knowledgebase has a meta-layer of information defining the structure and classification of the entities using a hierarchy model. The structure layer of the Wikipedia/DBpedia knowledgebase is defined through a set of categories which we generally call *topics*. Topics are abstraction concepts or classes which encircle a set of entities in a group. As an analogy for the relationship between topics and entities, consider the relationship between classes of objects and specific instances of those classes e.g. *Physicists* is a class and *Albert Einstein* is an instance of that class. Therefore, *Physicists* can be a topic and *Albert Einstein* an entity which belongs to that topic. Moreover, it is possible that topics share entities and, therefore, have overlapping domains. For instance, the topics of *Lawyers* and *Politicians* have many entities in common since there are many lawyers who enter politics. Lastly, the range and specificity of topics may also vary greatly based on the domain of a knowledgebase. Wikipedia, being a large generic knowledgebase, offers an expansive set of topics, describing every domain of knowledge and ranging from the very precise, such as “Fictional secret agents and spies”, to the very general, such as “Information”.

An interesting notion about entities and topics is that there are certain entities and topics which are identified with the same label. For instance, “Physics” and “University of Waterloo” are both labels of an entity as well as a topic. Later, we see that such entities and topics have a certain relationship in datasets, but, it is valuable to see why such phenomenon exists. The answer is that, while such entities and topics are identified similarly, their definitions differ fundamentally. An entity represents a certain specific concept for which it is labeled, while a topic defines a certain domain for a group of entities belonging to that domain. For example, the entity of “University of Waterloo” identifies the academic institution while the topic of “University of Waterloo” specifies the group of all the entities that belong to this topic. Clearly, the entity of “University of Waterloo” belongs to this topic but the following entities also belong to the “University of Waterloo” topic: “University of Waterloo Faculty of Engineering”, “St. Paul’s University College”, and “Waterloo Warriors”. Sometimes the distinction between such entities and their topics is difficult to discern e.g. “Physics” as an entity represents the science itself while “Physics” as a topic represents the category of this science within the realm of all sciences. One good way to mentally distinguish such cases is to think of entities as standalone concepts while topics are containers of entities which also belong to a hierarchy of other topics. For instance, “Physics” as a topic is a sub-category of “Physical Sciences” and a super-category of “Thermodynamics”.

In order to have a clear designation of entities, topics, and their relation, we need to formulate the knowledgebase using these concepts. The knowledgebase core (KB) is defined by the following:

$$KB \mapsto \{E, T, P, R\} \tag{3.1}$$

In this equation, *E* represents the entities, *T* represents the topics, *P* represents the predicates, and *R* represents the set of all possible relations between *E* and *T*. The following shows how elements of the set *R* are defined:

$$R = \{R_p \mid p \in P\} \tag{3.2}$$

$$R_p = \{(s, p, o) \mid \mathrm{s}, o \in \{E, T\}\} \tag{3.3}$$

In the above equations, R_p defines various relations defined by sets of triples of subjects (*s*), predicates (*p*), and objects (*o*). Subjects and objects can be either entities or topics while predicates originate from available relations and properties in the datasets. For instance, we will see that wikilink relations from the pagelinks dataset provide a *LinksTo* relation between entities. Such relations might be demonstrated as below (short format):

$$R_{LinksTo} = \{(e_1, LinksTo, e_2) \mid (e_1, e_2 \in E) \wedge (e_1 \neq e_2)\} \tag{3.4}$$

Moreover, if and only if a relation R_p exists between any given subject *s* and object *o*, can we assert that the logical proposition of *p(s,o)* is true. Equation (3.5) demonstrates such as assertion in general terms. As a case in point, let us assume entity e_1 has a wikilink to entity e_2 i.e. $(e_1, LinksTo, e_2) \in R_{LinksTo}$. Therefore, we can assert that $LinksTo(e_1, e_2)$ is true. It is also very important to note the upcoming relation definitions are all only providing necessity conditions and not all triples conforming to these conditions would have the defined relation. The sources of relation triples are various datasets.

$$(s, p, o) \in R_p \Leftrightarrow p(s,o) \equiv True \tag{3.5}$$

Having the knowledgebase and the concept of entities and topics defined, we can delve into the sources of these concepts and how they are related to each other and one another. The next three sub-sections will describe entity-to-entity, topic-to-topic, and entity-to-topic relationships. They show the datasets used to specify such relationships and how these relationships create entity and topic graphs.

3.5.1 Entity Graph

The dataset used in the knowledgebase to represent entities is clearly the titles dataset. This is a trivial choice as the titles dataset is the collection of the identifier of the encyclopedia articles which, by definition above, identifies the entities. As shown before, the titles dataset has many different types of connections to both itself and other datasets. The portion of connections of the titles dataset which connects the entities to other entities is the base of the *Entity Graph*.

The entity graph is the graph created by the collection of titles as its nodes and the connections between titles as its edges. The entity graph is a directed graph which means the edges are not symmetrical. Moreover, Figure 3.7 shows four different types of connections from titles to other titles (connections to literals are not of interest here). Table 3.3 shows these connections, their source datasets, predicate URIs, and equivalent relations in knowledgebase.

Table 3.3: Entity graph edge types and specifications.

Connection	Dataset	Predicate URI	Relation
Redirection	Redirects	dbpedia-owl:wikiPageRedirects	RedirectsTo
Disambiguation	Disambiguation Links	dbpedia-owl:wikiPageDisambiguates	DisambiguatesTo
Wikilink	Wikipedia Pagelinks	dbpedia-owl:wikiPageWikiLink	LinksTo
Property	Ontology Infobox Properties	dbpedia-owl:*Property_Label*	HasProperty

The first two types of connections, redirection and disambiguation, serve to specify the equivalent titles in the entity graph. These two types are used in the entity detection algorithm to match main titles to input keywords. However, these connections do not add extra value to semantic analysis and their use is mainly limited to input processing. The formal definitions of the relations these two connections generate are as follows:

$$R_{RedirectssTo} = \left\{ \left(e_1, RedirectsTo, e_2\right) \middle| \begin{array}{l} \left(e_1, e_2 \in E\right) \wedge \\ \left(e_1 \neq e_2\right) \wedge \\ \left(\neg \exists e_3 \in E : RedirectsTo(e_1, e_3) \wedge \left(e_2 \neq e_3\right)\right) \wedge \\ MainTitle\,(e_2) \end{array} \right\} \tag{3.6}$$

$$R_{DisambiguatesTo} = \left\{ \left(e_1, DisambiguatesTo, e_2\right) \middle| \begin{array}{l} \left(e_1, e_2 \in E\right) \wedge \\ \left(e_1 \neq e_2\right) \wedge \\ MainTitle\,(e_2) \end{array} \right\} \tag{3.7}$$

$$MainTitle(e) \equiv True \Leftrightarrow \left(\begin{array}{l} \left(\neg \exists e' \in E : RedirectsTo(e, e') \equiv True\right) \wedge \\ \left(\neg \exists e'' \in E : DisambiguatesTo(e, e'') \equiv True\right) \end{array} \right) \tag{3.8}$$

Equation (3.6) defines the *RedirectsTo* relation by stating that: one entity (e_1) can redirect to another entity (e_2), one entity cannot redirect to itself ($e_1 \neq e_2$), and one entity cannot redirect to more than one entity. Equation (3.7) defines *DisambiguatesTo* relation by stating that: one entity (e_1) can disambiguate to another entity (e_2) and one entity cannot disambiguate to itself ($e_1 \neq e_2$). Both of these equations also state that the object entity (e_2) of the triple should be a main title i.e. it should not be a redirect or a disambiguation title as defined by Equation (3.8). Note that *MainTitle* relation has no formal definition and is determined only by titles dataset. Therefore, it is accepted as ground truth.

The third type of connection is the largest part of the entity graph: wikilinks. Table 3.2 indicates that there are about 146 million edges of wikilink type. Therefore, wikilinks are statistically the best source of semantics between entities. Equation (3.9) defines the wikilink relation called *LinksTo*. This equation is the extended version of Equation (3.4) and it states that an entity can link to a different entity (not itself) and both entities should be main titles.

$$R_{LinksTo} = \left\{ \left(e_1, LinksTo, e_2\right) \middle| \begin{array}{l} \left(e_1, e_2 \in E\right) \wedge \left(e_1 \neq e_2\right) \wedge \\ MainTitle(e_1) \wedge \\ MainTitle(e_2) \end{array} \right\} \tag{3.9}$$

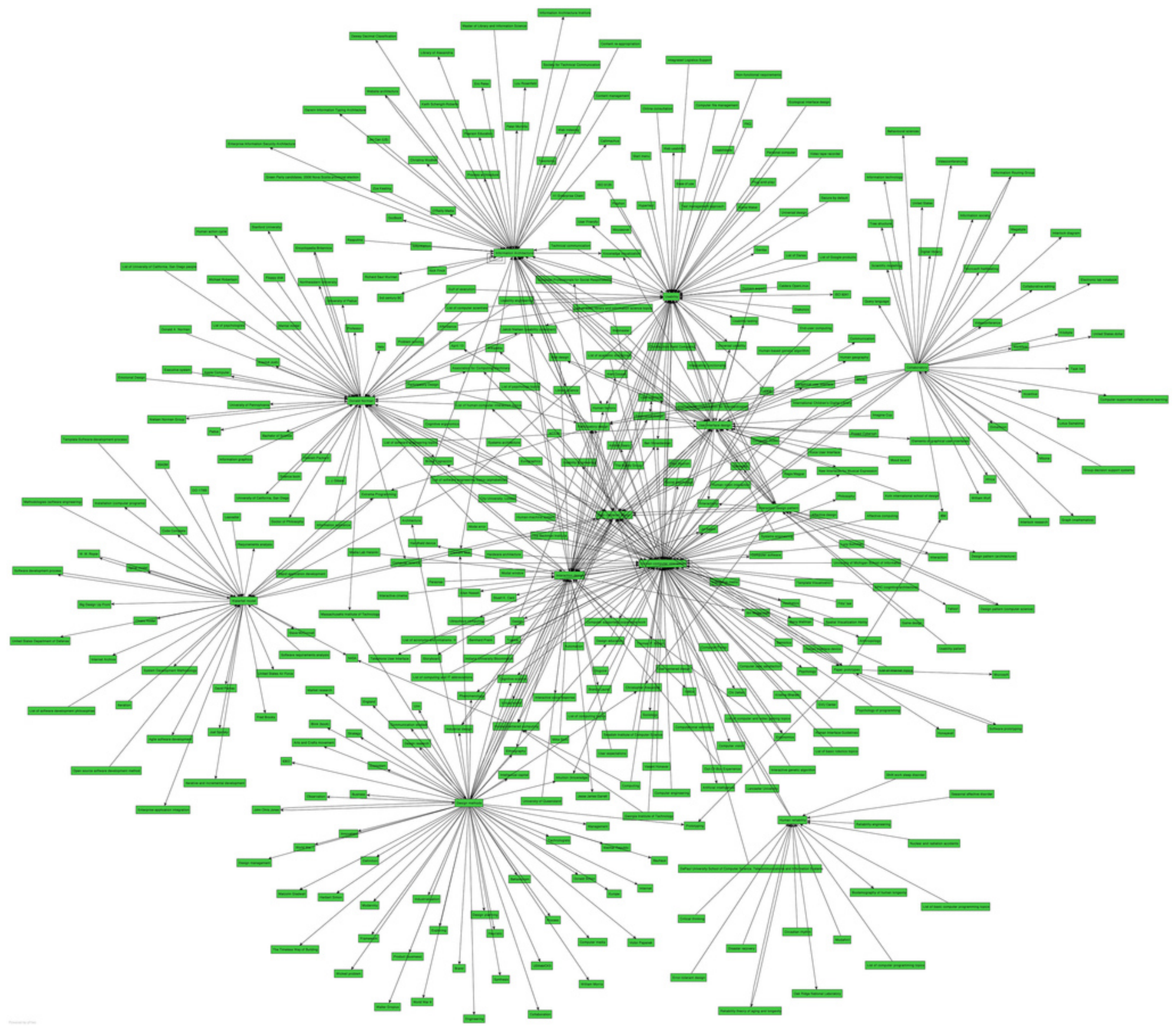


Figure 3.8: A segment of the entity graph featuring wikilink connections (adopted from [90]).

Figure 3.8 illustrates a small segment of the entity graph which features wikilink types of connections. The central node is the “Human-computer interaction” entity and the rest are links up to 2 levels deep.

The final type of the entity graph edges is an infobox property. As discussed earlier, some of the infobox property triples have other entities as their objects. Therefore, property triples provide another type of entity-to-entity connection. The major difference between these connections and wikilinks is that the predicate part of the triple is also a variable indicating what property of the subject entity the object entity is identifying. That is why the URI in Table 3.3 will vary for each different property. Having access to different properties for each entity semantically enriches any analysis and is very useful for natural language processing tasks such as question answering. However, for the scope of this research, entity graph traversal methods only differentiate between a property connection and a wikilink connection in general and do not distinguish between different types of properties. Therefore, instead of defining one relation per property, a generic relation called *HasProperty* is defined to represent infobox property triples.

$$R_{HasProperty} = \left\{ \left(e_1, HasProperty, e_2\right) \middle| \begin{array}{l} \left(e_1, e_2 \in E\right) \wedge \left(e_1 \neq e_2\right) \wedge \\ MainTitle(e_1) \wedge \\ MainTitle(e_2) \end{array} \right\} \tag{3.10}$$

The wikilinks and properties provide similar connections in the entity graph. Using a generic predicate such as *HasProperty* to define property triples causes its formal definition to resemble the wikilinks definition closely. Therefore, it seems as if properties are equal to wikilinks format-wise and only provide extra connections. This, however, is not semantically the case. There are two major differences between these two types of edges in the entity graph which will be reflected in semantic analysis later on. The first major distinction is that almost all entities have at least one wikilink while according to Table 3.2, only about half of the main titles have infobox information and consequently possible properties. The number of wikilinks of each entity is also usually much higher than number of its properties (if there are any properties for that entity). The second contrast between the two types of connections is that the properties are much more finely chosen than wikilinks. The properties are based on templates which feature cherry-picked features of entities which are most closely related to the topic of the entity article while wikilink can be chosen much more freely and, therefore, lack the desired semantic accuracy. In other words, one might always call a property closely relevant to the entity which is not the case for all wikilinks. To summarize, wikilinks provide a higher volume of information and are better for collective semantic analysis whereas properties are of lower quantity but are much more precise in terms of semantic relevance.

The elements discussed above constitute the entity graph of the semantic graph. We will see in Section 3.6 how the input is broken into possible entities and afterwards in Section 3.7, how the entity graph provides a collection of candidate entities for aggregated operations to determine entities most similar to the input entities. Such analysis, however, is usually only one part of the semantic analysis. Another major process on the semantic graph uses the topic graph by traversing its hierarchy and determining semantic distances between topics and between entities. The next section will elaborate on the definition and the structure of the topic graph.

3.5.2 Topic Graph

The dataset used to represent topics in the knowledgebase is the category (SKOS) dataset. As mentioned before, this dataset provides a massive set of categories which are connected by the hyponym relations between them, where a child category has an “is-a” relationship to its parents. Such relationships create a hierarchical structure which we will call the *Topic Graph*. Each category in the topic graph represents a node and the relationships between the category and its parents represent the edges of the topic graph. It is notable that in the topic graph each node (category) can have multiple parents as well as multiple children. Having multiple children is trivial since any topic can have multiple sub-topics about more specific concepts e.g. Engineering can have sub-topics such as Computer Engineering, Mechanical Engineering, and Chemical Engineering. The possibility of multiple parents, however, derives from topics which either belong to more than one concept or act as nexus points for different concepts. For instance, “Golf” as a topic can be thought of as a professional sport or a recreational hobby and, therefore, can belong to both parent topics: “Professional sports” and “Outdoor recreation”. On the other hand, the topic “American physicists” is a topic which connects American scientist in any field with physicists from any nationality and, therefore, it can belong to both parent topics: “American scientists” and “Physicists by nationality”. Figure 3.9 depicts a partial view of Wikipedia’s category hierarchy with arrows pointing downward to sub-categories.

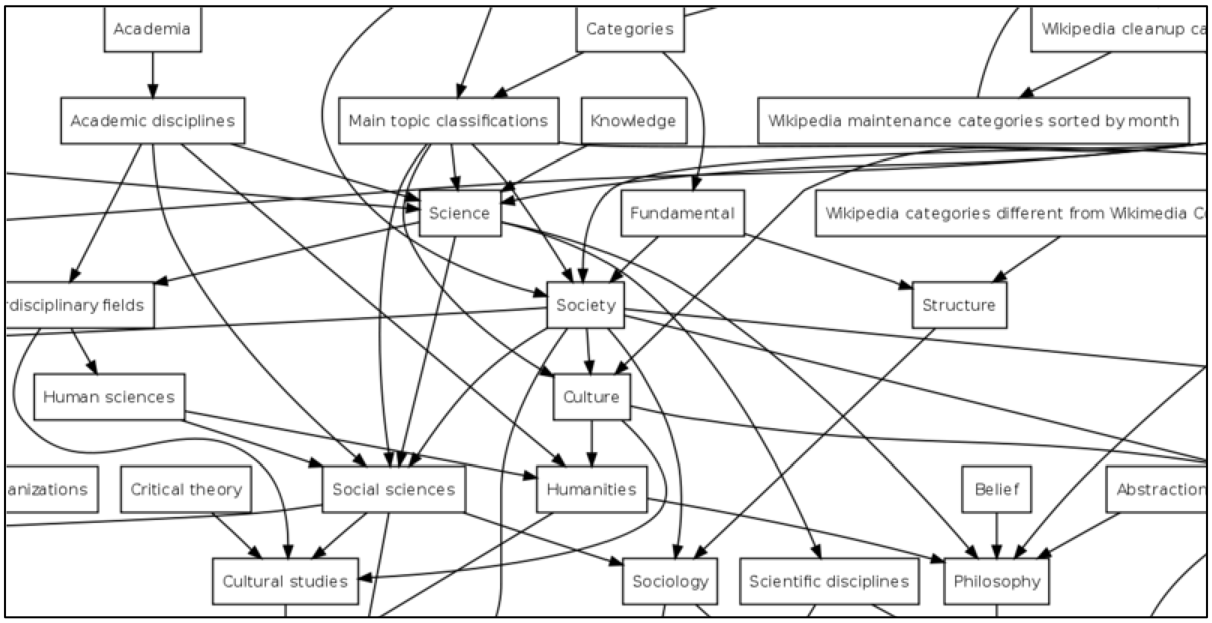


Figure 3.9: Partial view of Wikipedia's category system (adopted from [91]).

As shown in Figure 3.7 earlier, the category dataset provides a skos:broader predicate to create hyponym relations between categories. Table 3.4 demonstrates the specifications of the edges and the relationships of the topic graph while Equation (3.11) and Equation (3.12) formally define *IsSubCategoryOf* and *IsSuperCategoryOf* relations between topics. The former relationship shows that the subject of the triple is a child of the object of the triple while the latter relationship represents the opposite i.e. the subject is a parent of the object.

Table 3.4: Topic graph edge and relation specifications.

Connection	Dataset	Predicate URI	Relation
Broader Topic	Category (SKOS)	skos:broader	IsSubCategoryOf
Narrower Topic	Category (SKOS)	skos:narrower (inverse of skos:broader)	IsSuperCategoryOf

$$R_{IsSubCategoryOf} = \left\{ (t_1, IsSubCategoryOf, t_2) \middle| \begin{array}{l} (t_1, t_2 \in T) \wedge (t_1 \neq t_2) \wedge \\ IsSubCategoryOf(t_2, t_1) \equiv False \end{array} \right\} \quad (3.11)$$

$$R_{IsSuperCategoryOf} = \{ (t_1, IsSuperCategoryOf, t_2) \mid IsSubCategoryOf(t_2, t_1) \equiv True \} \quad (3.12)$$

Note that although two connections are mentioned in Table 3.4, there is only one type of connection for the topic graph edges. The two relationships mentioned above are using the same dataset as the source and, practically, express the same relation in different ways. The only difference is that one is the inverse of the other and is defined for the ease of use. The URI used in the dataset is only skos:broader and to find the inverse relation, the subject and the object in the triple would be swapped. This would semantically be equal to keeping the subject and the object positions intact and change the URI to skos:narrower instead. Equation (3.11) defines the *IsSubCategoryOf* relation by specifying that if topic t_1 is a sub-category of topic t_2, then the parent topic t_2 cannot be the sub-category of the child topic t_1. A topic cannot be a sub-category of itself either. Equation (3.12) defines that a topic t_1 is the super-category of t_2, if and only if t_2 is a sub-category of t_1.

Even though the above formulation provides a definite hierarchy of hyponym relationships, the graph is not strictly hierarchic as noted earlier: there exist shortcuts in the connections (i.e. starting from one child category and going up two different paths of different lengths to reach the same parent category) as well as loops (i.e. starting from one child category and going up a path to reach the same child category again). These anomalies are introduced as a result of the incremental generation of

Wikipedia's structure by human authors and the lack of a systematic enforcement of strict hierarchy rules. Therefore, a pre-processing method may need to be applied to the topic graph so that a strict version can be generated and inconsistencies can be pruned from the topic graph. Figure 3.10 shows some examples of such anomalies. The red dashed lines show the extra routes and loops. Notice how the "Society" category becomes a sub-category of itself indirectly by a 3-level loop which violates the last condition of Equation (3.11). Also, in this figure, the arrows are upward showing skos:broader connections in contrast to Figure 3.9 in which downward arrows represent skos:narrower connections.

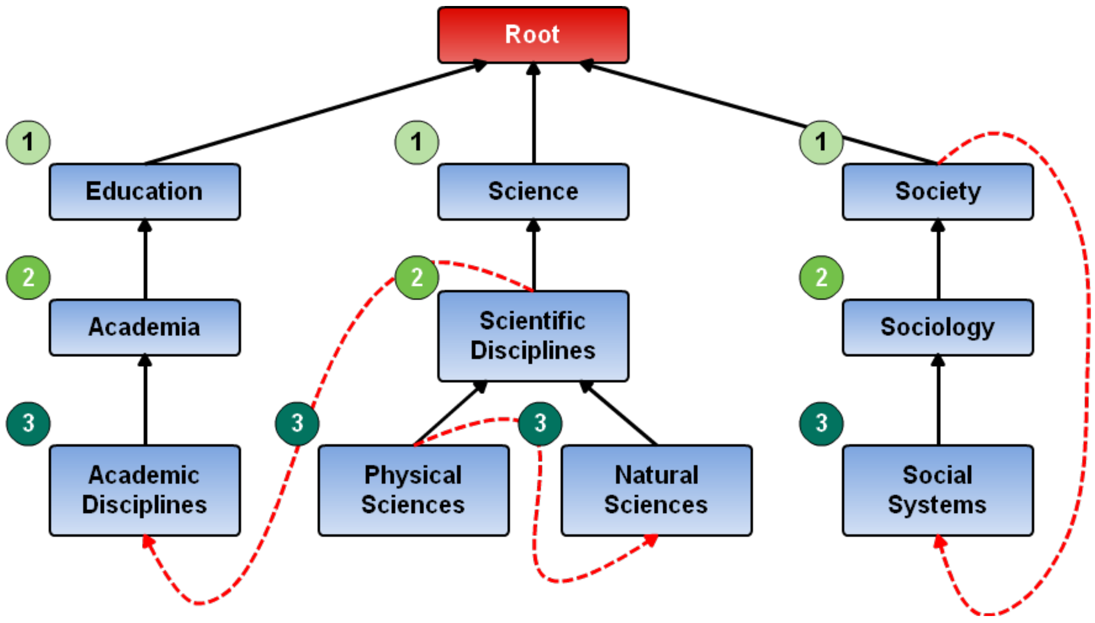


Figure 3.10: Examples of topic graph anomalies.

Pruning and pre-processing of the topic graph needs a number of steps performed. There is a faction of categories which is for maintenance, internal use, or organization purposes. There are also a few categories which have no parents or are disconnected from the main topic graph. The topic graph pre-processing algorithm should handle all these cases as well as inconsistent links and edges.

To perform such tasks, a leveling procedure is applied on the topic graph assigning each category with a level number which indicates the shortest path length from the root to that category. Figure 3.11 shows the complete topic graph pre-processing algorithm. The algorithm is quite simple. It receives one or more nodes as the root of the graph, assigns them a level of zero, and then assigns the immediate children of the root nodes a level 1 number. This means all immediate children of root nodes are 1 step away from the root. Then all the immediate children of level 1 nodes which have no level number assigned yet are found and given a level 2 number. The next step will search for the level 2 nodes' immediate children and assigns them a level 3 number and so on. This process is repeated until no unassigned children are found. Afterwards, the remaining unassigned nodes will be the ones with no parents, in disconnected sub-graphs, or out of the sub-graph identified by the given root nodes. These unassigned nodes will be labeled as *Pruned*. Moreover, any edge that indicates a node with a lower level number is a sub-category of another node with equal or higher level number is deemed an anomaly and, therefore, will be marked as *Pruned*.

```
Input: Desired root nodes of the processed topic graph
1. Assign a Level 0 to root nodes
2. L ← 1
3. Current_Level_Nodes ← All sub-categories of (L-1) level nodes
   with no level number assigned.
4. Assign level L to Current_Level_Nodes.
5. L ← L + 1
6. If Current_Level_Nodes is not empty go to step 3.
7. Mark all unassigned nodes as Pruned.
8. Mark any broadening edges connecting a node of level X to any
   node of level Y where X <= Y as Pruned.
```

Figure 3.11: Topic graph pre-processing algorithm.

As a result of the above algorithm, each row in Figure 3.10 corresponds to one level. For instance, “Science” will have a level of 1 and both “Physical Sciences” and “Natural Sciences” will get a level 3 number. In this case, according to step 8 of the algorithm, the edge connecting “Physical Sciences” as a sub-category of “Natural Sciences” is marked *Pruned* because both of them have equal level number. Similarly, the edge from “Scientific Disciplines” to “Academic Disciplines” makes a level 2 node the sub-category of a level 3 node which, according to step 8, earns the link a *Pruned* label. Moreover, note that when the children of level 3 are found, “Physical Sciences” will not be considered to get a level 4 number as the child of “Natural Sciences” because it has already been assigned a level 3 number.

Another important issue is the choice of root nodes for the pre-processing algorithm. The real root of Wikipedia's category system is called “Contents”. This category, however, has a lot of maintenance and internal sub-categories which are not of interest for the semantic graph. Therefore, the root nodes given to the pre-processing algorithm as input are “Main topic classifications” and “Fundamental categories”. These two are grandchildren of the “Contents” category and fathers of the semantic portion of Wikipedia’s category graph which we are interested in. Using these two as root nodes automatically renders the undesirable children of the “Contents” category unassigned and, therefore, marked as Pruned.

The leveling procedure explained in the Figure 3.11 algorithm can be formally expressed by Equation (3.13)[3] which expresses each topic (*t*) has a natural number called level (*lvl*). This topic is a sub-category of another topic with a level one lower (*lvl* – 1) and there exists no other topic which is a super-category of (*t*) and has a level number lower than (*lvl* – 1) i.e. there are no other shorter paths to root available.

$$R_{HasLevel} = \left\{ \left(t, HasLevel, lvl\right) \middle| \begin{array}{l} \left(t \in T \right) \wedge \\ \left(lvl \in N \right) \wedge \\ \left(\begin{array}{l} \exists t' \in T : IsSubCategoryOf\,(t, t') \wedge \\ HasLevel(t', lvl - 1) \end{array} \right) \wedge \\ \left(\begin{array}{l} \neg\exists t'' \in T : IsSubCategoryOf\,(t, t'') \wedge \\ HasLevel(t'', L) \wedge \left(L < (lvl - 1) \right) \end{array} \right) \end{array} \right\} \tag{3.13}$$

Table 3.5 presents some statistics after performing the topic graph pre-processing algorithm. The first two rows show that 125,780 categories were pruned after the processing. The following rows show the frequency of categories in each level. It is interesting to note the diamond shape which the growth/shrinkage trend has created. There are a few categories at the root which grow rapidly to 150955 categories wide at level 6 and then it starts to decrease to only 36 categories for the last level. The reason for growth is the rapid branching of the child categories at higher levels which broaden the width. But as we go deeper, there would be less and less categories and we will just see leaf nodes and that is why the number shrinks back. This diamond shape will be the shape of most sub-graphs of the topic graph as the trend is self-repeating in smaller dimensions as well.

An important notion about the above leveling algorithm is that it imposes certain limitations on the structure of the topic graph. These limitations may cause loss of generality and, therefore, reduction in performance for certain applications by decreasing the possible paths between topics and/or removing certain topics. It is possible to bypass applying this algorithm and work on the original

[3] *N* represents the natural numbers and *R* represents the real numbers.

topic graph. However, to avoid anomalies such as loops or topics with no parents, the traversal algorithm operating on the unpruned topic graph should keep a record of visited nodes and steer away when it comes in contact with an already visited node. This approach is more relaxed than leveling algorithm and allows better semantic exploration while it still recovers from wrong paths; however, it requires more space and more time for traversal compared to leveling algorithm. The extra space is needed for keeping a list of already visited nodes in each traversal and the extra time is needed for exploring all the adjacent nodes in each step compared to the leveling approach which only explores the ones with only lower levels (or only higher levels depending on the application).

With the topic graph defined, we have a solid semantic ontology structure for analysis. The major advantage of the topic graph is its hierarchy compared to the network structure of the entity graph. The topic graph hierarchy enables meaningful traversal methods for semantic analysis while the entity graph analysis mostly relies on aggregate clustering methods as described in Section 3.7. However, there is one last crucial element in the semantic graph to be discussed before any semantic analysis can be performed: the Entity-Topic connection layer.

Table 3.5: Processed topic graph statistics.

Topic Graph Source	**Wikipedia Dump October 2010 – DBpedia 3.6**
Total Categories	632607
Pruned Categories	506827
Level 0	2
Level 1	25
Level 2	933
Level 3	12152
Level 4	51850
Level 5	122269
Level 6	150955
Level 7	96065
Level 8	48982
Level 9	12311
Level 10	6448
Level 11	3118
Level 12	1067
Level 13	540
Level 14	74
Level 15	36

3.5.3 Entity-Topic Connection Layer

To perform a semantic analysis, there should be a link between the entity graph and the topic graph. As defined in Section 3.2, the processing portion of the methodology starts by detecting the entities from the input and then traversing the semantic graph. For some cases the traversal is limited to the entity graph. For many other cases, the analysis moves from entities to topics and performs topic graph traversal instead of, or in parallel with, entity graph traversal. To achieve this transition from entities to topics, a connection layer is needed between the entity graph and the topic graph which we call the Entity-Topic (ET) connection layer.

As mentioned before, each entity belongs to one or more categories and each category may have many entities attached. Therefore, there exists a many-to-many relation between entities and topics. The dataset providing the relation is the article-categories dataset. While this dataset only provides dcterms:subject URI linking entities to topics, by swapping the subject and object of its triples, we can define relations for both directions similar to the topic graph relations. Table 3.6 presents *HasCategory* and *IsCategoryOf* relations and Equation (3.14) and Equation (3.15) provide formal definition of these relations respectively.

Table 3.6: Entity-Topic connection layer relations.

Connection	Dataset	Predicate URI	Relation
Entity to Topic	Article Categories	dcterms:subject	HasCategory
Topic to Entity	Article Categories	*inverse of* dcterms:subject	IsCategoryOf

$$R_{HasCategory} = \left\{ (e, HasCategory, t) \,\middle|\, \begin{array}{l} (e \in E) \wedge (t \in T) \wedge \\ MainTitle(e) \end{array} \right\} \tag{3.14}$$

$$R_{IsCategoryOf} = \left\{ (t, IsCategoryOf, e) \,\middle|\, \begin{array}{l} (t \in T) \wedge (e \in E) \wedge \\ HasCategory(e,t) \equiv True \end{array} \right\} \tag{3.15}$$

Note that, by definition, only entities which are main titles can have categories. To fetch the categories of all other titles, they should first be traversed to their proper main titles. Furthermore, since *IsCategoryOf* is just a formal inverse of the *HasCategory* relation, it does not have physically corresponding triples in the article-categories dataset and it just uses the inversed triples. The inverse URI in Table 3.6 is mentioned solely for semantic correctness of the definition.

It is notable that some (not all) categories have a main article page assigned to them. This translates to some topics having a main entity among other entity connections. The main entity of a topic usually has the same label as the topic. Such cases are already discussed in Section 3.5. However, having the same label is not always the case and there is no other distinct indication of such relations between entities and topics either. Therefore, we will treat all connections between entities and topics in the connection layer equally.

Having defined the entity graph, the topic graph, and the connection layer, we can consider the semantic graph a complete ontology structure for the purpose of our methodology. The entity graph acts as a container for knowledge items. However, it does not provide much structure. The topic graph, on the other hand, provides a hierarchical structure suitable for semantic analysis. Consequently, the connection layer connects the elements of the entity graph to the elements of the topic graph providing indirect structure for the entity graph by placing entities on the fringes of the topic graph. In other words, the topic graph acts as the meta-data for the entity graph.

Figure 3.12 depicts a simplistic illustration of the structure of the entire semantic graph. As mentioned before, the topic graph resembles a diamond shape after being processed and assigned levels. The topic graph is at the top providing structure. The entity graph, however, is just a network of entity nodes interconnected through various uni- or bi-directional edges and it is shown as a circle of knowledge items at the bottom. Then each entity is connected to one or more topics shown by dashed lines. These connections constitute the Entity-Topic connection layer.

Lastly, it should be noted that the infobox types dataset, which provides the main type/class of some entities, is not used in the semantic graph. Types are generally good indicators of what an entity is in essence and they might be useful in some question answering problems. However, since there is no structure or hierarchy in the types dataset and also due to a limited number of types, meaningful semantic analysis cannot be applied on a variety of applications using types.

3.5.4 Challenges

While the semantic graph defined above is well-structured, there exist several issues with both the entity graph and the topic graph. Here, some of these issues are discussed and some clues on how to tackle them are provided. While it is not necessary to address these issues in every application, it is attempted to provide proper algorithms later on in order to address them in proper problem context.

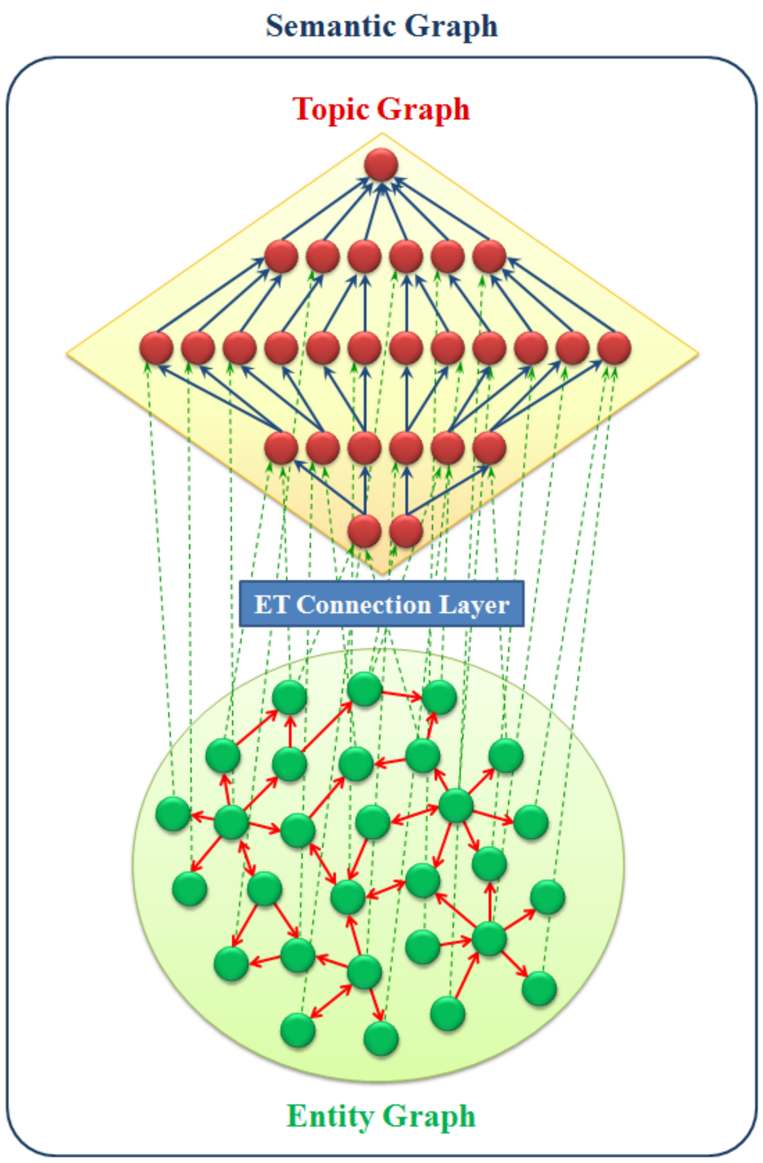


Figure 3.12: Semantic graph structure.

The entity graph, as mentioned, provides two main types of semantic connections between entities: properties and wikilinks. The advantages of these two types have already been discussed: wikilinks have a higher quantity and hence better statistical characteristics while properties have been cherry-picked as important features of an entity and semantically more relevant. If an approach needs to consider a wikilink and a property and rank them, it is usually the case that properties get a higher weight in scoring as we will observe in Section 3.7. However, the entity graph does not provide any distinction between wikilinks or between properties for each separate type i.e. edges are not weighted.

In other words, given an entity, we would not know, based on that entity itself, which of its wikilinks are more important than the other. Similarly, we cannot easily determine which of the properties of a single entity are more important. This means, we might need to invent some sort of weighting method to rank wikilinks and properties of an entity. The ranking of these connections, nonetheless, depends on the application. An importance measure should be defined for each application and then the links should be ranked based on that importance measure if needed. Furthermore, the weights of the entity graph edges are also dependant on the context. For instance, when the context is the personal life of a politician, the links to his/her spouse and children are more important. But, if the context is about the political life of the politician, the links to the offices held by him/her has more weight.

The same argument applies to the topic graph. There is no weight to the topic graph edges and, therefore, we do not know which of the parents or children of a given category are more important than the others. For instance, the topic "Toyota" belongs to many parent topics such as: "Car manufacturers of Japan", "Companies of Japan", and "1937 establishments in Japan". The edges in the graph have the same weight for all these three parents but one might suggest that the order above is a better indicator of the importance of each parent. But again, such ordering cannot be intuitively found for each topic and its connection. Moreover, similar to the entity graph, it is often the case that the weight of edges depends on the application and context and should be extracted dynamically.

Finally, the ET connection layer edges are also weightless and, therefore, there are no specific distinguishing factors between topics of a given entity. In this case, however, the types dataset might be useful to indicate the main topic of an entity but that would only indicate the topmost category for each entity and the rest of the topics would still remain on equal standings.

To address these issues in practice, we would need a set of entities and topics to determine the context and then perform the edge ranking using aggregation methods. These aggregations can be between entities, topics, or both. For instance, if a list of entities is given, we can find the most common topics between these entities to distinguish between all the topics of all the given entities.

In another possible approach, we can use approaches similar to TF-IDF. For instance, suppose we need to find more relevant parent topics of a certain given topic. If a parent topic is the parent of many other topics as well, it indicates that the parent is not closely related to the given topic while if a parent has only the given topic as the child, it would probably be very relevant to its child.

Some of the above issues, and the approaches to tackle them, will be discussed later on in more detail. However, before one can perform any semantic analysis over the semantic graph, the entities of the input to the system should be known as they are the entry points to the semantic graph. The next section will discuss how to detect the entities from the input.

3.6 Entity Detection

Entities are elements of the knowledgebase defining our world model. This means every textual statement regarded as an input to this methodology is assumed to consist of a number of entities connected to each other by natural language connectors. Therefore, for any further processing of the input, it should be broken into its building blocks i.e. entities. This process is called *entity detection*. Once we have a list of entities, we are able to find the connections between these entities and other entities or topics, perform semantic analysis, and finally, map them to the output for each specific problem. In this section, the base algorithm template for entity detection is described and its parameters are discussed. Later on, the exact algorithms for specific applications are laid out.

As indicated in Section 3.2, the input is some kind of textual object and to start the processing, the entities should be extracted from the text. The process of mapping text to entities is a common requirement for most other works. For instance, works of [92-94] explain methods for different services provided by the Wikipedia Miner project [95]. These services include automatic annotation of a text document with Wikipedia links which is called *wikifying* the document. Another method is to find the extent to which two text items are related to each other. Another work is to find main topics of a document using Wikipedia article titles. The main challenge of all of these methods is to map parts of the input text to Wikipedia article titles which we call entities. Another series of works are [96-98] which are the basis of a project called Wikitology. These works try to address a range of different use cases comprehensively explained in [99]. Some of these use cases are document concept prediction, entity linking to knowledgebase entities, interpreting tables, and document description. Again, all of these works rely on matching Wikipedia articles to the input documents.

The methodology presented here treats the input text as a collection of entities glued together by lingual connectors such as verbs, prepositions, articles, and so on. The approach is to break the text and extract these entities. However, finding the exact match is not always straightforward. Some entities can be expressed in different forms e.g. “American history”, “History of United States”, “U.S. History”. While the redirects might cover some of these variations, not all entities can be matched perfectly and there is always a possibility of extra or missing words in the input. Trying to compensate for extra or missing words by being flexible in matching, on the other hand, causes false positive matches. Therefore, the strategy of entity detection here is to match text to multiple entities in parallel and score them based on their closeness. Moving forward, all the processes will be performed on a set or subset of the candidate entities using aggregation operators. For example, if there is a word called *phoenix* in the input, it can be matched to many entities as shown in Section 3.3. The entity detection algorithm considers all the variations. As the process goes ahead and extracts more entities from the input, the analysis tries to recognize the context of the input and then it might prefer the mythical bird sense of the word phoenix to the city of Phoenix sense, for example.

```
Input: A textual object
   (I)   Processing Division
           • Apply text breaking
           • Remove stop words
  (II)   Search Division
           • Search entities for matching keywords
           • Store search result information: location and statistics
 (III)   Scoring Division
           • Apply word featuring weights
           • Apply penalty weights
           • Apply ordering weights
  (IV)   Equivalency Division
           • Insert and score redirects
           • Insert and score disambiguated entities
   (V)   Production Division
           • Determine the number of final candidate entities.
```

Figure 3.13: Entity detection algorithm.

Figure 3.13 shows the generic template for the entity detection algorithm. This template has a number of divisions. Some of these divisions might be customized for different problems. This means, firstly, a division might be completely ignored, completely applied, or partially applied.

Secondly, the method used for each division might be different based on the problem being solved. This section will explain each division and iterates through possibilities for customization of the divisions and the reasons behind why each customization can be applied.

3.6.1 Processing Division

Processing division has the responsibility of preparing the input text for search. The first preparation task is to manage the size of the text which might vary in different applications from a few words to a large document. In order to efficiently extract entities, a large textual object must be broken into smaller components especially if the words of an entity do not have the same order in the input text. Moreover, finding entities in a large document as a whole will not benefit from the context of the text if processed at once, but if the input is streamed in instead, after finding a few entities, the context can be recognized and used in spotting correct entities later on. Also, if the words from the input constituting an entity are not in the correct location or there is a disjoint, the entity detection must consider a range of words to find the correct entity. If this range is too large, the possibility of detecting false positives increases. As a case in point, consider the following text: “Phoenix is the capital and the largest city of the U.S. state of Arizona, a state which has a long border with Mexico”. To find the correct entity for “Phoenix, Arizona”, the entity detection algorithm should count for out of order and disjoint possibilities. But if the input is not broken into smaller components, some false positives such as “Mexico City”, “Capital of Mexico” can also be detected.

Text breaking methods used in this methodology are dependent on the application. Here, three possible text breaking methods are introduced. Later in Section 3.8, we will discuss appropriate breaking methods for different applications. The main three text break methods are:

1. Using text internal separators
2. Fixed word window with overlap: *Window(n, o)*
3. Variable *n*-gram window

The first method can be applied on text inputs that do provide certain separators. For instance, a document written in normal language uses sentences separated by certain punctuation marks such as a period (.) and a semicolon (;). This method is very simple and works for ordinary text inputs with comparatively short sentences.

The second method is also a simple method but provides more flexibility which makes it a better choice for more applications. In this method a fixed length moving window of *n* words is considered to break the text. The window is applied on the beginning of the text input which produces the first component. Then, the window is moved ahead enough so that it covers the last *o* words of the first component plus $n - o$ new words. To clarify, suppose that $n = 10$ and $o = 4$. The first window will cover the words 1 to 10. The second window starts from word 7 and covers till word 16. This means words 7, 8, 9, and 10 are repeated as the overlaps of two components and word 11 to 16 would be the $10 - 4 = 6$ new words of the second component. The reason for having an overlap between windows is to keep the coherence and the flow of the text. The choice of *n* and *o* depends on the nature of text and can be set empirically. The number of overlap words can be set to zero if needed. Furthermore, it is possible to use variable-sized windows and overlaps but a justified strategy must be designed for this type of selection. This method is appropriate for most text breaking applications, especially if unordered matching is required.

The third method is specifically for applications that require strict ordered matching. This method starts from the first word and searches for matching candidate entities. Then, it searches for the bi-gram combination of first and second word matching entities. If there are any bi-gram available, it will search for a tri-gram made up of the first, the second, and the third word and repeats this until it finds all the possible *n*-grams in matching entities. Once done, it moves one word ahead and starting from the second word, it repeats the incremental *n*-gram search again. Finally, the method finds all the matching candidate entities that have an *n*-gram from the input. Since this is a brute force algorithm and might take a long time, the length of *n*-grams can be limited to a maximum value. As this method searches for a combination of words rather than individual words, the order of the input will be maintained in the resulting entities. A more detailed example of this algorithm is presented in Section 4.3 for speech recognition applications.

The second task of processing division is to clean the input. This is often done by removing stop words i.e. articles and common pronouns and verbs such as: a, the, he, I, go, come, etc. Performing this task again depends on the type of the application. For instance, for WQC applications, this will be performed as web queries are usually not complete sentences and, therefore, stop words do not count as keywords. For speech recognition application, on the other hand, we will preserve the stop words as we are looking for exact matches.

3.6.2 Search Division

Search division is the common division among all variations of the entity detection algorithm i.e. no entity detection algorithm bypasses search division. It simply tries to find the words from the input in the list of entities. In fact, search division returns a list of all entities that feature at least one of the words of the current input component. This means the search might be performed on broken components separately and, therefore, on multiple occasions. The type of search might also be different based on ordering. If order does not matter, all entities that feature one or more words are found. Otherwise, only entities that feature a sequence of input words will be returned. As mentioned above, the latter search approach is the case in *n*-gram text breaking method. Each *n*-gram is a broken text component and it is searched so that an exact match is found in entities.

Besides searching and fetching entities, this division might also record certain information about the search. This information can be about the statistics of the matching, such as the number of input words found in an entity, the total words of the entity, and the length of the longest ordered sequence of input words found in an entity. It might also be about the location of found words in an entity. Such information can be used in subsequent divisions. Similar to earlier customization choices, inclusion or exclusion of stored information depends on the type of application and scoring methods applied.

3.6.3 Scoring Division

Scoring is the crucial division of the entity detection algorithm in terms of the accuracy and the performance of the algorithm. As stated earlier, for each input component, many entities would be detected as candidate entities. Each of these candidate entities will have a score indicating how much the algorithm believes the entity has been correctly detected. The scores will then be used to rank the entities accordingly. Consequently, the scores are the parameters determining how accurate the entity detection was achieved.

The score equation used in this methodology is comprised of three separate weights: word featuring weight, penalty (or proportional importance) weight, and ordering weight. The word featuring weight indicates how many of the input component words are in the entity. The more words are in an entity, the better match it would be. Penalty weight accounts for the extra portion of the entity not covered by the input. This penalty can take on different forms. We will introduce three possibilities in the formalization sub-section. The rationale for penalty weight is, clearly, that a lower penalty is desirable. All of the forms of the penalty weight are, however, defined as the proportion of the part of the input covered by the entity when compared to the entire entity. Any part of the entity not covered by any words of the input will result in a mismatch and, subsequently, a decrease in the final score. Finally, the ordering weight is an optional added incentive for the entities which feature words with the same order as the input. This part is only applied when search division considers no necessary order in entity matching; the algorithm prefers to reward entities with matching order over the ones without matching order. The details of the scoring mechanism will be discussed below shortly.

3.6.4 Equivalency Division

After finding and scoring candidate entities, we need to make sure that they are the main entry points to the entity graph. This means, if the candidate entities are among redirects and disambiguation titles, we need to find the main titles and insert them into the list of candidate entities. The task is very easy for redirects, if any of the entities has a redirect relation, the entity it redirects to will be inserted into the list. The newly inserted entity will inherit the score of the original entity as it is representing the same equivalent entity.

For disambiguation entities, there are two possible methods. The first method, similar to redirects, will insert all the disambiguated entities into the list and assign them the same score as their original entity. The second method is to insert all the disambiguated entities into the list but recalculate their score based on the same scoring strategy as scoring division. The rationale of the second method lies within the way disambiguation is handled. As mentioned in Section 3.3, disambiguated entities usually have the same label as the original entity with their sense annotated to them e.g. “Phoenix” will be “Phoenix (mythology)”, “Phoenix, Arizona”, and so on. Therefore, a recalculation of the score can be a good choice considering the possibility of the extra annotation being in the input component. For instance, if the input text is “Phoenix is the largest city of Arizona” and searching is considering out of order entity matching, “Phoenix, Arizona” is a perfect match compared to “Phoenix (mythology)” which has an extra word not matching any of the input words. Moreover, this second method of handling disambiguation equivalents can be combined into the search division. Since the disambiguated entities are main titles and already in the entity graph, if the original disambiguation entities are ignored in the search, we can be confident that the annotated entities will be automatically searched and scored without the need to find the equivalency.

3.6.5 Production Division

Once the candidate entities list is ranked and finalized, the production division decides what portion of the candidate list can be used further in the semantic analysis. This is done via applying a threshold over the scores and cutting off the entities with scores below the specified threshold. The threshold can be based on the rank as well as the score. It might also be fixed or variable. The following shows all four possibilities:

- **Fixed score threshold**: All the entities with score equal or above the specified threshold will be selected and the rest will be discarded.
- **Fixed rank threshold**: All the entities equal or above the specified rank will be selected and the rest will be discarded e.g. top 10 candidate entities will be selected.
- **Variable score threshold**: All the entities with a score equal or above a certain percentage of the highest entity score will be selected and the rest will be discarded. For instance, let us assume that the topmost candidate entity has a score of 4.0. A variable score threshold of 60% would select all the entities with score equal or above 4.0*60% = 2.4.
- **Variable rank threshold**: All the entities with a rank within a certain top percentage of the total number of candidate entities will be selected and the rest will be discarded. For instance, let us assume there are 50 candidate entities in total. A variable rank threshold of 20% will select the top 50*20% = 10 ranks.

Note that ranking can be done in two ways: regular ranking with ties (e.g. if there are 4 ties with rank 3 the next entity will start at rank 7 skipping the ranks of 4, 5, and 6) and dense ranking with no tie gaps (e.g. 4 ties with rank 3 will all have the same rank and the next entity gets the next immediate rank of 4).

3.6.6 Formalization

Formalizing the entity detection algorithm is not a straightforward task as the different nature of applications dictates various setups, structures, and formats. For instance, it will be shown that in the speech recognition problem that the input text is two or three parallel sentences unlike other problems with sequential word order. Yet, an attempt has been made to provide a generic view of the algorithm in formal language to better understand each division. Once the basic method is established, the transition to specific formats is easy to understand. The following demonstrates different divisions and their steps in the formal format and then tries to explain what each equation means.

The first step is to define the input text. Figure 3.13 defines the input as any textual object. To be a bit more specific, the input is defined as a set of words. These words might be in a specific order or might be just a bag of words. In some cases, such as text prediction, some of the words might not be complete i.e. there may be a few starting letters of the word. In general, we call the smallest unit of the input text a word. Equation (3.16) defines the input.

$$Input \mapsto \{W, L, S\} \tag{3.16}$$

In this equation, *W* is the set of words, *L* is the set locations for each word in *W*, and *S* indicates the separators if available. The exact definition of *L* and *S* depends on the structure of the input. The following equations define a number of different possibilities:

Bag of words (no order):
$$\begin{aligned} L &= \{\ \} \\ S &= \{\ \} \end{aligned} \tag{3.17}$$

A sequence of words:
$$\begin{aligned} L &= \{(w_i, i) \mid (w_i \in W) \wedge (i \in N)\} \\ S &= \{\ \} \end{aligned} \tag{3.18}$$

A sequence of sentences:
$$\begin{aligned} L &= \{(w_i, i) \mid (w_i \in W) \wedge (i \in N)\} \\ S &= \left\{ (w_i, s_i) \,\middle|\, \begin{array}{l} (w_i \in W) \wedge \\ (s_i \in \{".", ";", "!", "?"\}) \end{array} \right\} \end{aligned} \tag{3.19}$$

Parallel sentences(same size):
$$\begin{aligned} L &= \left\{ (w_{i,j}, (i, j)) \,\middle|\, \begin{array}{l} (w_{i,j} \in W) \wedge ((i, j) \in N) \wedge \\ (1 \le i \le I) \wedge (1 \le j \le J) \end{array} \right\} \\ S &= \left\{ (w_{I,j}, s_j) \,\middle|\, \begin{array}{l} (w_{I,j} \in W) \wedge (j \in N) \\ \wedge (1 \le j \le J) \\ \wedge (s_j \in \{".", ";", "!", "?"\}) \end{array} \right\} \end{aligned} \tag{3.20}$$

Equation (3.17) defines no location and no separators which means there is no word order in the input. A web query fits this definition since web queries are a set of words which are not necessary in the correct order. Equation (3.18) indicates a sequence of words e.g. one sentence with words indexed as w_1, w_2, w_3, …. Equation (3.19) is a sequence of sentences e.g. a text document. The words have a sequential order but there are certain separators breaking sentences at certain positions. Equation

(3.20) shows the format used in the speech recognition problem where there are two or three sets of sentences with an equal number of words in each sentence. In this equation, I indicates the total number of words in each sentence and J indicates the number of sentences. Each sentence ends with a separator in this format which means no other separators are in the middle of the sentences (note the capital I in $w_{I,j}$).

The first part of the entity detection algorithm, which uses the above input definition, is the processing division. As stated, the processing division breaks the input into smaller components and cleans the stop words based on the problem definition. Since both tasks are optional, they may or may not be performed. Equation (3.21) shows the result of processing division.

$$Comp_n = \left\{ \{K_n, L_{K_n}, S_{K_n}\} \middle| \begin{array}{l} (K_n \subset W) \wedge (L_{K_n} \subset L) \wedge (S_{K_n} \subset S) \\ \wedge (\forall (k,i) \in L_{K_n} : k \in K_n) \\ \wedge (\forall (k,s) \in S_{K_n} : k \in K_n) \end{array} \right\} \tag{3.21}$$

In the above equation, $Comp_n$ represents one sample component which is a subset of the input created by the text breaking algorithm. In this format, K_n would be a subset of input words, W. The reason for changing the variable from W to K is due to the stop word cleaning process. Regardless of performing the cleaning task, we consider the resulting <u>w</u>ords as <u>k</u>eywords hence the letter K. L_{Kn} and S_{Kn} will in turn represent the subset of locations and separators for those keywords included in the $Comp_n$ (if any were found). The following are two examples of the processing division tasks. The first example indicates an input with 5 words, no order, and no separators. No text breaking was performed but the cleaning task removes word 3:

$$\begin{array}{l} W = \{w_1, w_2, w_3, w_4, w_5\}, L = \{\ \}, S = \{\ \} \\ \rightarrow Comp_1 = \{K_1, L_{K_1}, S_{K_1}\} \\ \rightarrow K_1 = \{k_1, k_2, k_4, k_5\} = \{k_i | (k_i = w_i) \wedge (w_i \in W) \wedge (i \in \{1,2,4,5\})\} \\ \rightarrow L_{K_1} = \{\ \}, S_{K_1} = \{\ \} \end{array} \tag{3.22}$$

The second example shows an input with 10 words with sequential order and no separators. A 7-word window, 3-word overlap text breaking algorithm (*Window(7,3)*) was performed but no cleaning was done:

$$\begin{aligned}
&W = \{w_i \mid 1 \le i \le 10\}, L = \{(w_i, i) \mid (w_i \in W) \wedge (1 \le i \in 10)\}, S = \{\ \} \\
&\rightarrow Comp_1 = \{K_1, L_{K_1}, S_{K_1}\}, Comp_2 = \{K_2, L_{K_2}, S_{K_2}\} \\
&\rightarrow K_1 = \{k_i \mid (k_i = w_i) \wedge (w_i \in W) \wedge (1 \le i \le 7)\} \\
&\rightarrow K_2 = \{k_i \mid (k_i = w_i) \wedge (w_i \in W) \wedge (4 \le i \le 10)\} \\
&\rightarrow L_{K_1} = \{(k_i, i) \mid (k_i = w_i) \wedge (w_i \in W) \wedge (1 \le i \le 7)\}, S_{K_1} = \{\ \} \\
&\rightarrow L_{K_2} = \{(k_i, i) \mid (k_i = w_i) \wedge (w_i \in W) \wedge (4 \le i \le 10)\}, S_{K_2} = \{\ \}
\end{aligned} \tag{3.23}$$

Having specified input components, search division tries to find all the entities in the semantics graph that feature at least one of the keywords of each component. The set of candidate entities for *Comp*$_n$ is called *E*$_n$ and it consists of candidate entities (e_c) and some information about them ($INFO_c$) used for scoring. Each candidate entity (e_c) in this set *contains* at least one keyword (k_i) from the keywords of the corresponding component (K_n). Equation (3.24) states the formal format of the candidate entities of each component.

$$E_n = \left\{ (e_c, INFO_c) \;\middle|\; \begin{array}{l} (e_c \in E) \wedge \\ (\exists\, k_i \in K_n : Contains(e_c, k_i)) \end{array} \right\} \tag{3.24}$$

The next step is scoring each candidate entity. As stated, the scoring division includes three weights for each entity score. The following will describe each of these three weights in formal format. However, it should be noted that the scoring division will apply the scores solely based on the nature of each entity and the keywords it contains. In some use cases, after performing semantic analysis on one component, an additional scoring mechanism will evaluate new scores based on semantic analysis and applies those over the scores presented here. The score for each candidate entity in E_n, called $S^n(e_c)$, is defined using three parameters: the word featuring weight (N_k), the penalty weight (P_k), and the optional ordering weight (O_k). The following equations describe these parameters in formal format:

$$S^n(e_c) = N_k \times P_k \left[\times O_k \right] \tag{3.25}$$

$$N_k = Count(k_i)\Big|_{Contains(e_c, k_i)} \tag{3.26}$$

$$\begin{cases} P_k = \dfrac{N_k}{N_e} \vee \\ P_k = \dfrac{C_k}{C_e} \vee \\ P_k = \dfrac{R_k}{R_e} = \dfrac{\sum F_k}{\sum F_e} \end{cases} \tag{3.27}$$

$$Contains(e_c, "k_i\ k_{i+1} \cdots k_j") \Rightarrow O_k = (j - i + 1) \tag{3.28}$$

All the parameters mentioned in the above equations, plus the locations of the input component keywords in the entities, are calculated by the search division and packaged inside the $INFO_c$ parameter of E_n.

$$INFO_c \mapsto \{N_k, N_e, C_k, N_e, F_k, F_e, O_k, Locations\} \tag{3.29}$$

The first parameter (N_k) indicates the number of keywords featured in the candidate entity (e_c). The higher the number of featured words is, the higher the score will be. The second parameter (P_k) accounts for the extra words of the entity not featured by any of input keywords. Equation (3.27) provides three possible definitions. The first is simply the proportion of keywords in the entity (N_k / N_e, where N_e is the total number of words in candidate entity e_c). The second is the proportion of characters in the entity that belong to keywords (C_k / C_e, where C_k is the total number of characters of the keywords featured in the entity and C_e is the total number of characters in candidate entity e_c). This metric assumes that longer keywords are more important; in the context of web queries, which are only a few words long [54, 100], it may be true that more emphasis was meant by the user on the longest most evident word in the query. The final measure of proportional importance is based on the word's inverted frequencies. It is computed as the sum of inverted frequencies of the keywords in the entity to the sum of inverted frequencies of all entity words (ΣF_k / ΣF_e), where the inverted frequency of a word w is computed as:

$$F_w = \ln(Num_E / Num_w) \tag{3.30}$$

In Equation (3.30), Num_E is the total number of entities in the semantic graph and Num_w is the number of entities featuring word w. It is, in essence, the IDF part of the classic TF-IDF equation: (N_w / N_e)ln(Num_E / Num_w), where N_w is the number of instances of word w in a specific entity (or more generally, a document) and N_e is the total number of words in that entity. The TF part (N_w / N_e) is ignored because it does not give a reliable result when dealing with short entities that only feature each word once or twice. This metric has been used successfully in the past in a classifier working on WQC problem using Wikipedia [62].

Lastly, the third parameter (O_k) is used when the entity detection wants to accept out-of-order keyword searches in entities but also wants to prefer the entities that match input order. In this case, the ordering weight would be the length of the longest sequence of keywords appearing in the entity label in correct order. Note that if the search mechanism already considers order in the search O_k will be equal to N_k and it would not have any effect on the scoring. Therefore, O_k is only used when out of order searching is performed.

To clarify the scoring mechanism, consider the input: "Geography of United States". After applying stop word removal (similar to Equation (3.22)), the keywords of the only input component is: K_1 = "Geography United States". Now consider the following entities:

$$\begin{aligned} e_1 &= \text{"United States"} \\ e_2 &= \text{"United States Geography"} \\ e_3 &= \text{"United States History"} \end{aligned} \tag{3.31}$$

Entity e_1 features $N_k = 2$ words of the K_1 and has $N_e = 2$ words in total. Entity e_2 features $N_k = 3$ words of the K_1 and has $N_e = 3$ words in total. Finally, entity e_3 features $N_k = 2$ words of the K_1 and has $N_e = 3$ words in total. Considering the first definition of penalty weight and using no ordering weight, the scores of each candidate entity would be:

$$\begin{aligned} S^1(e_1) &= N_k \times {}^{N_k}\!/_{N_e} = 2 \times {}^{2}\!/_{2} = 2.0 \\ S^1(e_2) &= N_k \times {}^{N_k}\!/_{N_e} = 3 \times {}^{3}\!/_{3} = 3.0 \\ S^1(e_3) &= N_k \times {}^{N_k}\!/_{N_e} = 2 \times {}^{2}\!/_{3} = 1.3 \end{aligned} \tag{3.32}$$

It is evident that e_2 would be the top candidate and e_1 and e_3 would follow. It is interesting to notice how penalty weight will penalize e_3 for having an extra word dropping it to the last position. If we have used ordering weight above, all of the scores would have been multiplied by 2 as "United States" is the longest sequence of words in K_1 that matches the three entities. However, let us compare the score of e_2 for input component K_1 and input component K_2 = "United States Geography" using ordering weight.

$$S^1(e_2) = N_k \times {}^{N_k}\!/\!_{N_e} \times O_k = 3 \times {}^{3}\!/\!_{3} \times 2 = 6.0$$
$$S^2(e_2) = N_k \times {}^{N_k}\!/\!_{N_e} \times O_k = 3 \times {}^{3}\!/\!_{3} \times 3 = 9.0 \tag{3.33}$$

Since all three words in K_2 match the order of candidate entity e_2, O_k would be equal to 3 hence the score of $S^2(e_2)$ is equal to 9 compared to $S^1(e_2)$ which is 6.

After performing the scoring, equivalency division simply adds redirects and disambiguation links. As explained, redirects copy the score while disambiguation links might recalculate scores based on their annotated sense disambiguation tags. $f(INFO_c)$ represents the recalculation of the score in the following equations which show the formal format of equivalency division:

$$E_n \leftarrow E_n \cup \{(e'_c, INFO_c) \mid ((e_c, INFO_c) \in E_n) \wedge RedirectsTo(e_c, e'_c)\} \tag{3.34}$$

$$E_n \leftarrow E_n \cup \{(e'_c, f(INFO_c)) \mid ((e_c, INFO_c) \in E_n) \wedge DisambiguatesTo(e_c, e'_c)\} \tag{3.35}$$

Finally, the production division will cut off the candidate entities for each input component which has a score less than a certain threshold. As explained, the cut can be performed on the score or the rank of entities and the thresholds could be constant or a percentage of top ranked candidate entities. The following equations show the two possible threshold methods:

$$E_n \leftarrow E_n - \{(e_c, INFO_c) \mid S^n(e_c) < Threshold\} \tag{3.36}$$

$$E_n \leftarrow E_n - \{(e_c, INFO_c) \mid Rank(S^n(e_c)) < Threshold\} \tag{3.37}$$

The above formalization equations conclude the entity detection algorithm description. The next section will describe the next step in processing which is semantic analysis of the candidate entities resulting from the entity detection algorithm explained in this section.

3.7 Semantic Analysis

The key element and final part of the proposed methodology is the semantic analysis. Once the input has been translated to a list of candidate entities by the entity detection algorithm, the entry points to the semantic graph are specified. This means an analysis algorithm can pick up the entry points and traverse the semantic graph to find relevant entities and topics based on the use case exposed to the methodology. For instance, in case of query topic identification, once the input query is mapped to candidate entities, the semantic analysis algorithm will use the ET connection layer to find corresponding topics of the candidate entities. It then traverses the topic graph so it can find the closest, most relevant target topic and, by doing so, identify the topic of the query. As another example, the semantic analysis algorithm can use the candidate entities to recognize the context of the input and use the context for word sense disambiguation; that is, based on the candidate entities, it can decide which entity best matches the incoming input. Section 3.8 will introduce a list of possible use cases where the semantic analysis can be used to provide a solution along with possible solutions using semantic analysis.

To perform any of the tasks mentioned in the above examples, the semantic analysis algorithm must traverse the semantic graph often starting from multiple entry points passing through multiple paths and converging on a possible target or targets. Such traversal can be varied based on many different parameters. One parameter is whether the traversal should be performed on the entity graph, the topic graph, or both. Another parameter is whether the direction of the traversal is important or not. The importance of the distance between entry points and targets in traversal paths over the number of paths can be another factor. The targets themselves are, in some applications, known and, in some other applications, unknown. In some cases, the semantic analysis and entity detection will repeatedly be performed in an alternative manner on different input components and they produce a progressive analysis instead of a snapshot analysis. The following sections will describe different concepts and algorithms for semantic analysis considering the above parameters and how to tackle each appropriately. Once these concepts and algorithms are explained, Section 3.8 will provide some real-world applications and will use different semantic analysis algorithms explained below to provide possible solution for each application.

3.7.1 Semantic Distance

The first and most fundamental concept to understand in semantic analysis is the *semantic distance* between elements of the semantic graph. Most of the applications handled by this methodology require that the entities or topics which are semantically relevant or close to input to be identified. Therefore, there should be a metric to determine how much two ontological concepts are close to each other. This metric is called the semantic distance.

The semantic distance, also known as the semantic similarity or semantic relatedness, might have various definitions in different situations [101]. Cosine similarity is one similarity metric used often in text mining applications and uses the term frequency of documents as the vector describing each document. The cosine of two vectors representing two documents will be calculated using their

Euclidean dot product. Cosine values near 1 mean very similar and values near 0 mean not so similar documents. Latent Semantic Analysis (LSA) [102, 103] is famous for using this method and metric.

Since the knowledgebase in this methodology is defined by the semantic graph, the semantic distance should be defined accordingly. The items defining the semantic distance in this structure are entities and topics and, therefore, the semantic distance definition is based on the literal distance between the nodes of the semantic graph i.e. entities and topics. The distance between nodes in the semantic graph is specified by the length of a path connecting one node to another. This path, however, is not unique. For most of the nodes in the semantic graph, there are countless paths connecting each pair. Each of these paths might have a different length and, therefore, producing numerous semantic distances for any given two nodes (and that is without considering the loops in the entity graph). The trivial way for handling this issue is to find the shortest path between the two nodes in question. Alas, the definition of the shortest path between two nodes is not unique either as it depends on many factors such as: types of source and target nodes, types of the connecting edges, the direction of the edges, and the semantic weight of the edges. To explain how each of these factors affects the definition of the semantic distance, the different definitions are classified using the types of the two nodes for which the distance is defined. The following provides the semantic distance definitions for topic-to-topic, entity-to-topic, and entity-to-entity nodes along with other deciding factors in each case.

3.7.1.1 Topic-to-Topic Distance

Determining the semantic distance between two topics is the most straightforward of all the definitions. The topic graph is the upper layer of the knowledgebase with a hyponymy structure. There is only one edge type between topics which is a hyponym relation showing the child is a sub-category (sub-topic) of its parent. Therefore, the semantic distance between two topics would simply be the length of the shortest path between the two topics in the topic graph. Equation (3.38) shows the topic-to-topic semantic distance ($Dist_{TT}$) using the recursive form. The description of this equation is that if any two topics t_1 and t_2 are directly connected as one being the sub-category or super-category of another, the semantic distance would be equal to 1. Otherwise, if there is another topic t such that t is directly connected to t_2, the distance would be the distance between t_1 and t plus 1 given that there is no other topic directly connected to t_2 with a lesser distance to t_1 compared to t i.e. the distance between t_1 and t_2 is the shortest path length.

$$Dist_{TT}(t_1,t_2)=Dist_{TT}(t_2,t_1)=\begin{cases}0\Leftrightarrow\left(\left(t_1,t_2\in T\right)\wedge\left(t_1=t_2\right)\right)\\ \vee\\ 1\Leftrightarrow\left(\left(t_1,t_2\in T\right)\wedge\begin{pmatrix}IsSubCategoryOf(t_1,t_2)\vee\\ IsSuperCategoryOf(t_1,t_2)\end{pmatrix}\right)\\ \vee\\ Dist_{TT}(t_1,t)+1\Leftrightarrow\\ \left(\begin{array}{l}\left(t_1,t_2,t\in T\right)\wedge\\ \left(\begin{array}{l}\begin{pmatrix}IsSubCategoryOf(t,t_2)\vee\\ IsSuperCategoryOf(t,t_2)\end{pmatrix}\wedge\\ \neg\exists\, t'\in T:\left(\begin{array}{l}Dist_{TT}(t_1,t')<Dist_{TT}(t_1,t)\\ \wedge\begin{pmatrix}IsSubCategoryOf(t',t_2)\vee\\ IsSuperCategoryOf(t',t_2)\end{pmatrix}\end{array}\right)\end{array}\right)\end{array}\right)\end{cases}\quad(3.38)$$

The above is a general definition describing topic-to-topic semantic distance applicable to most use cases. However, there is a possibility that some applications would need to find the semantic distance given a certain direction for traversal. In other words, the definition given above is a symmetrical definition causing $Dist_{TT}(t_1, t_2)$ to be equal to $Dist_{TT}(t_2, t_1)$. But if the application asserts that, for example, the distance direction from t_1 to t_2 should be strictly from child to parent (t_1 should be a sub-category of t_2), then the distance would only be defined between children and their ancestors in a one-way direction. Any two topics not related through ancestry i.e. related through siblings or cousins, or related through the reverse direction of ancestry will get a distance of infinity. For instance, if the assertion is that the direction should only be upwards from children to parents and assuming that t_1 is a sub-category of t_2, then the distance[4] from t_1 to t_2 would be equal to 1: $Dist^U_{TT}(t_1,t_2)=1$ and the reverse would be infinity: $Dist^U_{TT}(t_2,t_1)=\infty$. Now if t_3 is another child of t_2 which means t_1 and t_3 are siblings, the distances between t_1 and t_3 in both directions would be equal to infinity: $Dist^U_{TT}(t_1,t_3)=Dist^U_{TT}(t_3,t_1)=\infty$. Figure 3.14 shows some examples of upward distances. The same argument, albeit reversed, is true if the assertion was downward from parents to children. Equation (3.39) and Equation (3.40) will define the upward and downward topic-to-topic distances respectively. The downward distances are the exact inverse of the upward distances. Having defined the upward distance, the downward distance can be defined by swapping variables as indicated by

[4] $Dist^U_{TT}$ represents the upward distance and $Dist^D_{TT}$ represents the downward distance.

Equation (3.40) meaning the downward distance between t_1 and t_2 is equal to the upward distance between t_2 and t_1.

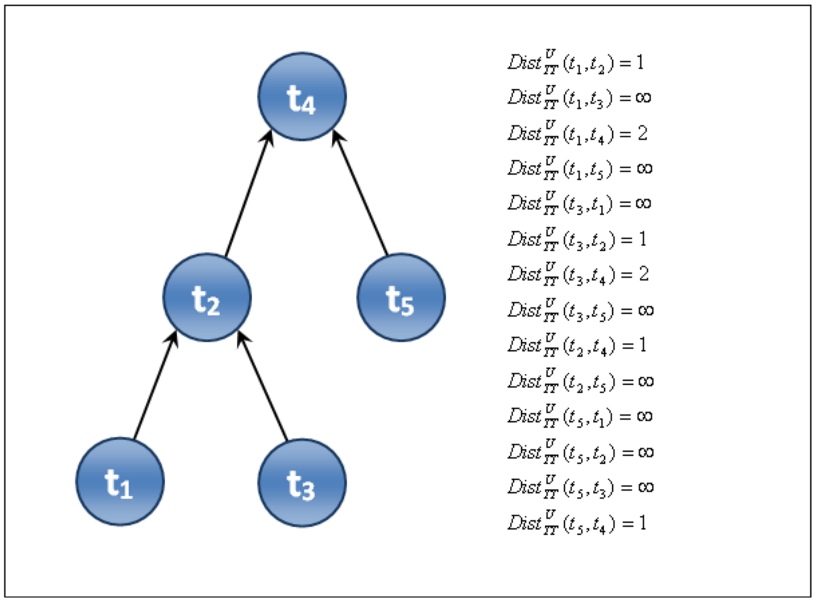

t4
t2
t5
t1
t3
$Dist_{TT}^{U}(t_1,t_2)=1$
$Dist_{TT}^{U}(t_1,t_3)=\infty$
$Dist_{TT}^{U}(t_1,t_4)=2$
$Dist_{TT}^{U}(t_1,t_5)=\infty$
$Dist_{TT}^{U}(t_3,t_1)=\infty$
$Dist_{TT}^{U}(t_3,t_2)=1$
$Dist_{TT}^{U}(t_3,t_4)=2$
$Dist_{TT}^{U}(t_3,t_5)=\infty$
$Dist_{TT}^{U}(t_2,t_4)=1$
$Dist_{TT}^{U}(t_2,t_5)=\infty$
$Dist_{TT}^{U}(t_5,t_1)=\infty$
$Dist_{TT}^{U}(t_5,t_2)=\infty$
$Dist_{TT}^{U}(t_5,t_3)=\infty$
$Dist_{TT}^{U}(t_5,t_4)=1$

Figure 3.14: Examples of upward topic-to-topic semantic distances.

$$Dist_{TT}^{U}(t_1,t_2)=\begin{cases}0 \Leftrightarrow \left(\left(t_1,t_2 \in T\right) \wedge \left(t_1=t_2\right)\right)\\ \vee \\ 1 \Leftrightarrow \left(\left(t_1,t_2 \in T\right) \wedge IsSubCategoryOf(t_1,t_2)\right)\\ \vee \\ \infty \Leftrightarrow \left(\left(t_1,t_2 \in T\right) \wedge IsSuperCategoryOf(t_1,t_2)\right)\\ \vee \\ Dist_{TT}^{U}(t_1,t)+Dist_{TT}^{U}(t,t_2) \Leftrightarrow \\ \left(\left(t_1,t_2,t \in T\right) \wedge \left(\begin{array}{l} IsSubCategoryOf(t_1,t) \wedge \\ \neg\exists\, t' \in T : \left(\left(\begin{array}{l} Dist_{TT}^{U}(t_1,t')+Dist_{TT}^{U}(t',t_2) < \\ Dist_{TT}^{U}(t_1,t)+Dist_{TT}^{U}(t,t_2) \end{array}\right) \wedge IsSubCategoryOf(t_1,t')\right) \end{array}\right)\right)\end{cases} \tag{3.39}$$

$$Dist_{TT}^{U}(t_1,t_2) \neq \infty \Rightarrow Dist_{TT}^{U}(t_2,t_1)=\infty$$

$$Dist_{TT}^{D}(t_1,t_2)=Dist_{TT}^{U}(t_2,t_1) \tag{3.40}$$

There might be an interest to define the semantic distance between two topics using the distance between entities of each topic. Such formulations vary greatly based on the definition and might be useful for certain applications but in the scope of this research, the semantic distance between two topics would be defined solely based on the topic graph.

3.7.1.2 Entity-to-Topic Distance

The meaning of the semantic distance between an entity and a topic is tricky. Entities and topics are naturally different concepts and, therefore, one might not be able to define the semantic distance between them. Since an entity might belong to a topic, we can interpret that as the entity being semantically very close to the topic it belongs to i.e. the minimum possible distance (*c*). Now to define the distance between an entity (*e*) and a topic (*t*) which it does not belong to, we can reduce the problem to finding the distance between topic (*t*) and another topic (t_e) to which entity (*e*) does belong. Since we have already defined topic-to-topic distance, we only need to add the minimum possible distance (*c*) to the topic-to-topic distance to calculate the distance between an entity and a topic. However, any given entity might belong to many different topics which means by the above definition, many distances can be calculated. The solution is simply to select a topic (t_e) for distance calculation which minimizes the distance compared to all other topics to which entity (*e*) belongs. Equation (3.41) shows the formalization of the above definition.

$$Dist_{ET}(e,t) = \begin{cases} c \Leftrightarrow (c \in R) \wedge (e \in E) \wedge (t \in T) \wedge HasCategory(e,t) \\ \vee \\ Dist_{TT}(t_e,t) + c \Leftrightarrow \\ \left(\begin{array}{l} (e \in E) \wedge (t_e, t \in T) \wedge \neg HasCategory(e,t) \wedge \\ \left(\begin{array}{l} HasCategory(e,t_e) \wedge \\ \neg\exists\, t' \in T : \left(\begin{array}{l} (Dist_{TT}(t',t) < Dist_{TT}(t_e,t)) \\ \wedge\, HasCategory(e,t') \end{array} \right) \end{array} \right) \end{array} \right) \end{cases} \quad (3.41)$$

In the above equation, *c* is a constant determining the proportional weight of an edge between an entity and its topic compared to an edge between two topics. Since topic-to-topic edges are valued at 1, selecting a value less than 1 for *c* will increase the topic graph dominance while a value larger than 1 for *c* will decrease the topic graph effect over an entity-to-topic semantic distance. Furthermore, if the entity-to-topic distance is only used in comparison with other entity-to-topic distances, the choice for *c* would be irrelevant as all distances would get the same value. In such cases, one might use a value of zero for *c* and reduce the problem to topic-to-topic distance comparison.

In this research, the distance from a topic to an entity is not applicable as entities are the entry points of the semantic graph. In the next subsection, however, we will see how entity-to-entity distance can be defined based on entity-to-topic distances and in such cases, the distance from a topic to an entity would be assumed the same as the distance from an entity to a topic.

3.7.1.3 Entity-to-Entity Distance

Defining a semantic distance between two entities is a complicated issue. Entities are connected to each other via different types of edges and different types of nodes. Entities are connected through the entity graph by means of redirects, disambiguation links, wikilinks, and properties. They are also connected through the topic graph by the use of hyponym relations between their topics. Therefore, to accurately define the entity-to-entity semantic distance, all of the above parameters should be taken into account. Equation (3.42) provides a generic definition considering all the different routes possible for connections between two entities.

$$Dist_{EE}(e_1,e_2) = Dist_{EE}(e_2,e_1) = \min\begin{pmatrix} Dist_{EE}^{R}(e_1,e_2), \\ Dist_{EE}^{D}(e_1,e_2), \\ \alpha \cdot Dist_{EE}^{P}(e_1,e_2), \\ \beta \cdot Dist_{EE}^{L}(e_1,e_2), \\ \theta \cdot Dist_{EE}^{T}(e_1,e_2) \end{pmatrix} \tag{3.42}$$

The distance, as defined in the above equation, tries to indicate the shortest path between two entities by considering the minimum of 5 possible route types. Each of these 5 routes provide symmetrical distances; thus, the order of arguments (e_1 and e_2) does not matter. The routes are, however, compared using different importance weights considering the essence of each route type. In the following, each route is described and a discussion is provided on how each of the weights is set.

The first two routes indicate redirects ($Dist_{EE}^{R}$) and disambiguation links ($Dist_{EE}^{D}$). These two types might only be available between certain entities. If an entity is a redirect or disambiguation of another entity, it means these entities are in fact equivalent and, therefore, there is no distance between them semantically. But if no such relation is available between two entities, redirects and disambiguation links do not contribute to a semantic distance definition; therefore, in case of absence, their distance would be considered as infinity. This will reduce the first two routes to binary functions resulting in either zero or infinity which means if they exist, they will supersede all other routes and causing the minimum to become zero; if they do not exist, they will not have any effect in the $Dist_{EE}$

definition as infinity would act as neutral element in a minimum aggregation function. Because of this zero versus infinity binary effect, adding an importance weight coefficient is unnecessary for these two routes. Equation (3.43) and Equation (3.44) demonstrate the above routes in a formal format.

$$Dist_{EE}^{R}(e_1,e_2)=\begin{cases} 0 \Leftrightarrow \left(e_1,e_2 \in E\right) \wedge \begin{pmatrix} \left(e_1 = e_2\right) \vee \\ RedirectsTo(e_1,e_2) \vee \\ RedirectsTo(e_2,e_1) \end{pmatrix} \\ \vee \\ \infty \Leftrightarrow \left(e_1,e_2 \in E\right) \wedge \begin{pmatrix} \left(e_1 \neq e_2\right) \wedge \\ \neg RedirectsTo(e_1,e_2) \wedge \\ \neg RedirectsTo(e_2,e_1) \end{pmatrix} \end{cases} \tag{3.43}$$

$$Dist_{EE}^{D}(e_1,e_2)=\begin{cases} 0 \Leftrightarrow \left(e_1,e_2 \in E\right) \wedge \begin{pmatrix} \left(e_1 = e_2\right) \vee \\ DisambiguatesTo(e_1,e_2) \vee \\ DisambiguatesTo(e_2,e_1) \end{pmatrix} \\ \vee \\ \infty \Leftrightarrow \left(e_1,e_2 \in E\right) \wedge \begin{pmatrix} \left(e_1 \neq e_2\right) \wedge \\ \neg DisambiguatesTo(e_1,e_2) \wedge \\ \neg DisambiguatesTo(e_2,e_1) \end{pmatrix} \end{cases} \tag{3.44}$$

The third route ($Dist_{EE}^{P}$) represents the property connections between entities. Since only half of the entities have infoboxes [86] and, therefore, might have properties, it is possible that not every two entities have a route between them which only consists of property edges. If such a route exists, each step would count as one unit towards the length of the route and, consequently, the distance. In practice, however, property routes are used in a step-by-step exploration method; this means that only entities immediately connected to a given entity through properties are selected. This implies the distance is usually one, but in such an exploration method, the deciding factor is the importance weight labeled as alpha (α) in Equation (3.42). The discussion on weights will follow shortly. Also note that the direction of the property link does not matter in this definition. Equation (3.45) provides a multistep definition of the property entity-to-entity distances.

$$Dist_{EE}^{P}(e_1,e_2)=\begin{cases}0 \Leftrightarrow \left(\left(e_1,e_2 \in E\right) \wedge \left(e_1 = e_2\right)\right) \\ \vee \\ 1 \Leftrightarrow \left(\left(e_1,e_2 \in E\right) \wedge \begin{pmatrix} HasProperty(e_1,e_2) \vee \\ HasProperty(e_2,e_1) \end{pmatrix}\right) \\ \vee \\ Dist_{EE}^{P}(e_1,e) + 1 \Leftrightarrow \\ \left(\begin{array}{l} \left(e_1,e_2,e \in E\right) \wedge \\ \left(\begin{array}{l} \begin{pmatrix} HasProperty(e,e_2) \vee \\ HasProperty(e_2,e) \end{pmatrix} \wedge \\ \neg\exists\, e' \in E : \left(\begin{array}{l} Dist_{EE}^{P}(e_1,e') < Dist_{EE}^{P}(e_1,e) \\ \wedge \begin{pmatrix} HasProperty(e',e_2) \vee \\ HasProperty(e_2,e') \end{pmatrix} \end{array}\right) \end{array}\right) \end{array}\right) \end{cases} \tag{3.45}$$

$$Dist_{EE}^{L}(e_1,e_2)=\begin{cases}0 \Leftrightarrow \left(\left(e_1,e_2 \in E\right) \wedge \left(e_1 = e_2\right)\right) \\ \vee \\ 1 \Leftrightarrow \left(\left(e_1,e_2 \in E\right) \wedge \begin{pmatrix} LinksTo(e_1,e_2) \vee \\ LinksTo(e_2,e_1) \end{pmatrix}\right) \\ \vee \\ Dist_{EE}^{L}(e_1,e) + 1 \Leftrightarrow \\ \left(\begin{array}{l} \left(e_1,e_2,e \in E\right) \wedge \\ \left(\begin{array}{l} \begin{pmatrix} LinksTo(e,e_2) \vee \\ LinksTo(e_2,e) \end{pmatrix} \wedge \\ \neg\exists\, e' \in E : \left(\begin{array}{l} Dist_{EE}^{L}(e_1,e') < Dist_{EE}^{L}(e_1,e) \\ \wedge \begin{pmatrix} LinksTo(e',e_2) \vee \\ LinksTo(e_2,e') \end{pmatrix} \end{array}\right) \end{array}\right) \end{array}\right) \end{cases} \tag{3.46}$$

The fourth route ($Dist_{EE}^{L}$) represents the wikilinks connections between entities. Every entity in the entity graph has at least one wikilink but it usually has more. Due to the high number of wikilinks, it is likely that any two given entities are connected through wikilinks. But it is also possible to find two entities without a multistep route consisting of wikilinks. Nevertheless, similar to property links, wikilinks are often used in step-by-step exploration methods which are only interested in the immediate connected entities. Therefore, the deciding factor again is the importance weight, labeled as beta (β) in Equation (3.42). Similar to properties, the direction of wikilinks does not matter in determining the semantic distance. Equation (3.46) provides a multistep definition of the wikilinks entity-to-entity distances.

The fifth and final route ($Dist_{EE}^{T}$) represents the connections between entities through the topic graph. Since each entity in the entity graph has at least one topic and since all topics are connected to each other through the topic graph, any two entities are connected to each other via at least one path (and possibly many more paths) through the topic graph. Therefore, it is always possible to find a semantic distance between two entities even if there are no paths in the entity graph to connect them. The trick is to find all the topics of each of the two entities and then find one topic from the first entity and another topic from the second entity which produce the shortest path possible among all other possibilities of topic-to-topic connections. Figure 3.15 illustrates an example. Suppose that e_1 has three topics (t_x, t_y, t_z) and e_2 has two topics (t_a, t_b). There are 6 possible connections between topics of e_1 and topics of e_2. In this example, the distance between t_y and t_a is the smallest, equal to 3. Therefore, the semantic distance between e_1 and e_2 would be equal to 3 + 2*c*. As mentioned in entity-to-topic distance, there is an edge with weight of constant *c* between any entity and its topics. Therefore, the final distance would always have a 2*c* add-on.

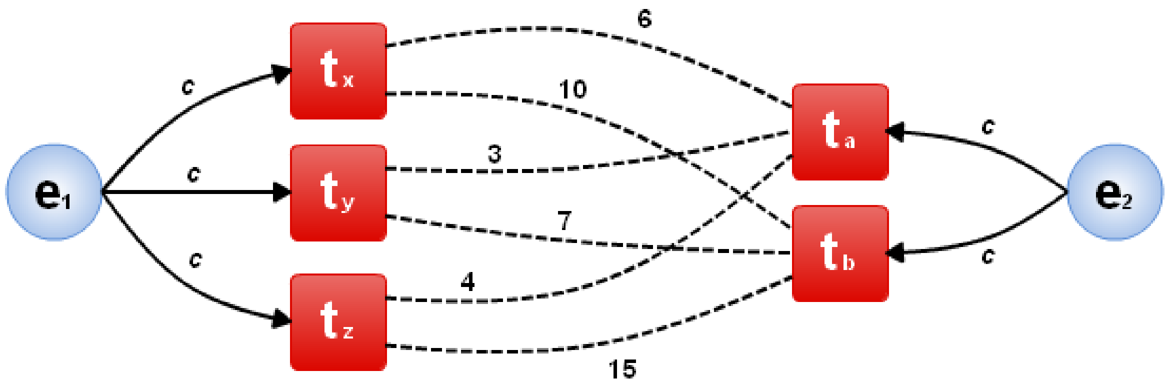


Figure 3.15: Entity-to-entity semantic distance through the topic graph.

While finding the semantic distance through the topic graph is always possible, similar to previous distances, we are mostly interested in entities which are immediately connected to the topics of a certain entity. For instance, in the above example, only entities which are connected to t_x, t_y, and t_z are of interest when dealing with e_1 as the starting point. This preference reduces the intermediate distances to zero which makes the deciding factor and final distance equal to 2c times the importance weight labeled as theta (θ) in Equation (3.42). Equation (3.47) expresses the formal format of entity-to-entity distance through the topic graph.

$$Dist_{EE}^{T}(e_1,e_2)=\begin{cases}0 \Leftrightarrow \left(\left(e_1,e_2\in E\right)\wedge\left(e_1=e_2\right)\right)\\ \vee \\ 2\cdot c \Leftrightarrow \left(\begin{array}{l}\left(c\in R\right)\wedge\left(e_1,e_2\in E\right)\wedge\\ \left(\exists\, t\in T:\left(\begin{array}{l}HasCategory(e_1,t)\wedge\\ HasCategory(e_2,t)\end{array}\right)\right)\end{array}\right)\\ \vee \\ Dist_{TT}(t_1,t_2)+2\cdot c \Leftrightarrow \\ \left(\begin{array}{l}\left(e_1,e_2\in E\right)\wedge\left(t_1,t_2\in T\right)\wedge\\ \left(\begin{array}{l}\left(\begin{array}{l}HasCategory(e_1,t_1)\wedge\\ HasCategory(e_2,t_2)\end{array}\right)\wedge\\ \neg\exists\, t_3,t_4\in T:\left(\begin{array}{l}Dist_{TT}(t_3,t_4)<Dist_{TT}(t_1,t_2)\\ \wedge\left(\begin{array}{l}HasCategory(e_1,t_3)\wedge\\ HasCategory(e_2,t_4)\end{array}\right)\end{array}\right)\end{array}\right)\end{array}\right)\end{cases} \tag{3.47}$$

As mentioned earlier, the final entity-to-entity semantic distance is defined based on the minimum of the 5 different routes explained above. However, there is also a possibility of defining routes which are a mix of the last three options. If the entities fall within the redirects and disambiguation links, they are basically equivalent and the distance would automatically be zero. However, considering the last three options, it is possible to define a route which uses different types of connections in its different steps. For instance, consider a route from the starting entity to a second entity through a wikilink, then to a third entity through a property link, and finally, to a fourth entity through a topic graph link. Considering such mixed format will produce numerous possible routes, finding the shortest path will then become a very computationally demanding task. To include such routes but at the same time simplify the algorithm, a step-by-step exploration algorithm is introduced. This algorithm will try to create the best mix-and-mash route through a greedy approach. Although the algorithm will be explained later, it is important to note how this approach affects the importance weighting scheme introduced in Equation (3.42). Setting the proper weights for α, β, and θ is based on how they are used in each step of the exploration algorithm. Figure 3.16 illustrates entities which are immediately connected to a certain entity (*e_1*) through different types and will appear in one step of the exploration algorithm. It also shows the importance weights for each type.

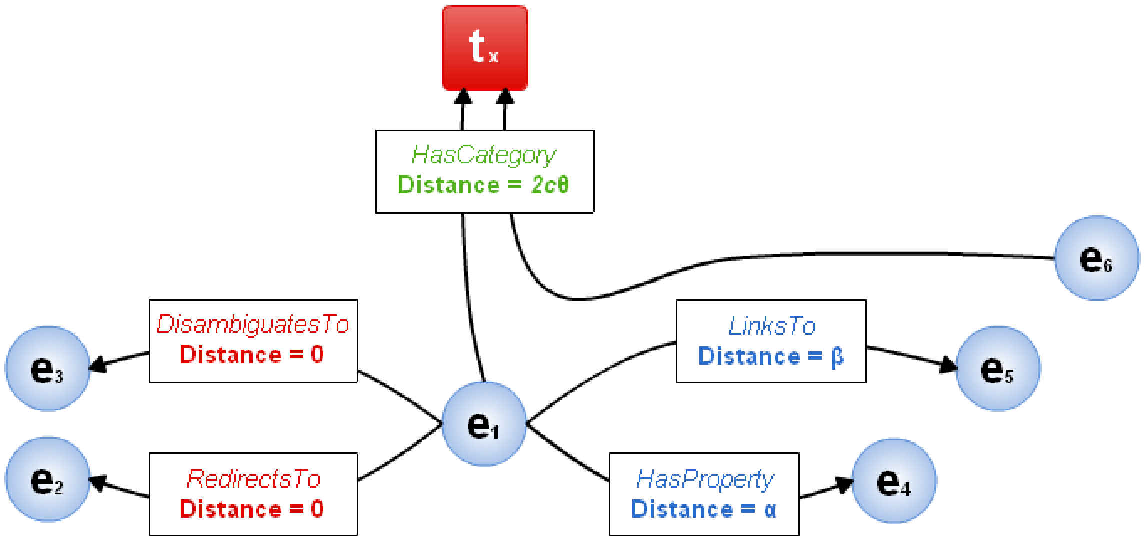


Figure 3.16: Entity-to-entity immediate connections through various types.

Entities e_2 and e_3 are equivalent and, therefore, have zero distance to e_1. Entities e_4 and e_5 are connected through the entity graph and have distance of α and β respectively. Lastly, entity e_6 is connected through the topic graph and has a distance of 2 × c × θ. The rationale behind setting proper values for α, β, and θ is based on the essence of each route. Since the properties are specified by Wikipedia authors as important links between two entities, they are considered the closest entities (which are not equivalents) to a given entity. Afterwards, wikilinks will be the closest ones. Since wikilinks do not have definite predicates, they might not be as accurate as properties but they are still closer concepts to an entity as they are directly connected compared to the entities connected through the topic graph. Therefore, one step distance through the topic graph should be weighted higher than wikilinks. Using the above rationale, proper values for α, β, and θ should satisfy the following condition:

$$\alpha \leq \beta \leq \theta \tag{3.48}$$

Note that the value of constant *c* cannot be zero since it causes all weights to become zero based on the above condition. A good value for *c* is in fact 0.5 because it will simplify the $Dist_{EE}^{T}$ distance for one step to θ (= 2 × 0.5 × θ). Moreover, based on the application, each of the three routes might be ignored when finding semantic distance. For instance, due to the nature of a problem, one might ignore using properties all together. Also, it is possible to select the same value for all the weights and make the semantic distance equal for all types (considering *c* = 0.5).

3.7.2 Semantic Distance Calculation Algorithms

Although a complete definition of semantic distance was presented, there was no discussion on how to practically calculate these distances. The calculation of the distance between two entities or two topics is not very complicated. However, the different setups dictated by the problem definition can affect the algorithm mechanism. Here, a number of different algorithms are briefly introduced and discussed. Each of these algorithms tackles a variation of the distance calculation based on a different setup.

3.7.2.1 Topic-to-Topic Generic Semantic Distance

In this case, the goal is to find the distance between any given two topics. To be generic, the goal is defined as to find the distances between two sets of topics instead of two individual topics. Figure 3.17 provides the full algorithm specification and presents the steps.

The algorithm starts by creating a set called *Calculated Distances* (*CD*). This set will have the distances between starting topics and destination topics and all the intermediate topics in between. It is initialized by the zero distances from starting topics to themselves. There is another variable defined as l which stores the current path length starting at zero.

Step 2 checks that if there is no destination topic (t_d) which has at least one missing distance to any of the starting topics, the algorithm is done and jumps to step 6. In other words, it checks to see if the distances between all starting topics and all destination topics have been calculated. Next in step 3, the algorithm tries to find the distances to all the new topics which are one step further from the topics it already calculated the distance of l for. This is done by gathering all the topics (t_n) with a distance of l from any of the starting topics (t_s) and then finding all of their sub- or super-category topics (t_m) whose distances to the starting topics are not already in *CD*. Such sub- or super-category topics will get a distance of $l + 1$ to the starting topics (t_s, t_m, $l + 1$). Then l is increased by one and the algorithm continues until all the distances are calculated.

Note that in this algorithm, the first element of the *CD* tuples is always a starting topic ($t_s \in TS$). When the algorithm stops, not only the distances to all destination topics are calculated for all starting topics, but all the intermediate topic distances have also been calculated which can be used in later processing.

```
Goal:      Given a set of starting topics ($TS$) and a set of
           destination topics ($TD$), find $Dist_{TT}(t_s,t_d)$ for any $t_s \in TS$
           and $t_d \in TD$.
Input:     $TS \subset T$: Set of starting topics
           $TD \subset T$: Set of destination topics
Output:    $Dist_{TT}(t_s,t_d)$ for all $t_s \in TS$ and $t_d \in TD$
1. $CD \leftarrow \{(t,t,0) \mid t \in TS\}$, $l \leftarrow 0$
2. If $\{t_d \mid (t_d \in TD) \wedge (\exists t_s \in TS : \neg\exists n \in N : (t_s,t_d,n) \in CD)\}$ is empty, go to step 6
3. $CD \leftarrow CD \cup \left\{ (t_s,t_m,l+1) \;\middle|\; (t_s,t_n,l) \in CD \wedge \begin{pmatrix} (\neg\exists\, n \in N : (t_s,t_m,n) \in CD) \wedge \\ \begin{pmatrix} IsSubCategoryOf(t_n,t_m) \vee \\ IsSuperCategoryOf(t_n,t_m) \end{pmatrix} \end{pmatrix} \right\}$
4. $l \leftarrow l+1$
5. Go to step 2
6. $\forall((t_s,t_d,l) \in CD) \wedge (t_s \in TS) \wedge (t_d \in TD)$ return $Dist_{TT}(t_s,t_d) = l$
```

Figure 3.17: Topic-to-topic generic semantic distance calculator algorithm.

3.7.2.2 Topic-to-Topic Upward and Strictly Upward Semantic Distance

Here, two variations of previous algorithm are presented. As mentioned earlier, in some applications, it is needed that the traversal direction in finding the semantic distance be limited to a certain course. Here, the algorithms for upward and strictly upward calculation are discussed which are very similar to the generic one. The only difference is that the upward algorithm only chooses topics which are parents of the topics at the current *l* distance. The strictly upward algorithm is the same as the upward algorithm with the added constraint that it employs the processed topic graph using levels concept defined in Equation (3.13). In the strictly upward algorithm, not only the next step topics should be parents of the current *l* distance topics, they should also have a lower level than their child topics. Note that in upward and strictly upward algorithms, it is possible that no upward path between a source topic and a destination topic exists. That is why there is an extra step in case of no change in *CD*, which jumps out of the loop and considers remaining distances to be infinity. Upward algorithm is specified by $Dist_{TT}^{U}(t_s, t_d)$ and strictly upward algorithm is specified by $Dist_{TT}^{SU}(t_s, t_d)$. We will observe that the strictly upward algorithm will be used in Parla application explained in Section 3.8. Figure 3.18 and Figure 3.19 present the specification and steps of upward and strictly upward algorithms respectively.

```
Goal:     Given a set of starting topics ($TS$) and a set of
          destination topics ($TD$), find $Dist_{TT}^{U}(t_s, t_d)$ for any $t_s \in TS$
          and $t_d \in TD$.
Input:    $TS \subset T$: Set of starting topics
          $TD \subset T$: Set of destination topics
Output:   $Dist_{TT}^{U}(t_s, t_d)$ for all $t_s \in TS$ and $t_d \in TD$
1. $CD \leftarrow \{(t, t, 0) \mid t \in TS\}$, $l \leftarrow 0$
2. If $\{t_d \mid (t_d \in TD) \wedge (\exists t_s \in TS : \neg\exists n \in N : (t_s, t_d, n) \in CD)\}$ is empty, go to step 7
3. $CD \leftarrow CD \bigcup \left\{ (t_s, t_m, l+1) \middle| (t_s, t_n, l) \in CD \wedge \begin{pmatrix} (\neg\exists\, n \in N : (t_s, t_m, n) \in CD) \wedge \\ IsSubCategoryOf(t_n, t_m) \end{pmatrix} \right\}$
4. If CD remains unchanged, go to step 7
5. $l \leftarrow l + 1$
6. Go to step 2
7. $\forall((t_s, t_d, l) \in CD) \wedge (t_s \in TS) \wedge (t_d \in TD)$ return $Dist_{TT}^{U}(t_s, t_d) = l$
```

Figure 3.18: Topic-to-topic upward semantic distance calculator algorithm.

```
Goal:     Given a set of starting topics ($TS$) and a set of
          destination topics ($TD$), find $Dist_{TT}^{SU}(t_s, t_d)$ for any $t_s \in TS$
          and $t_d \in TD$.
Input:    $TS \subset T$: Set of starting topics
          $TD \subset T$: Set of destination topics
Output:   $Dist_{TT}^{SU}(t_s, t_d)$ for all $t_s \in TS$ and $t_d \in TD$
```

```
1. $CD \leftarrow \{(t,t,0) \mid t \in TS\}$,  $l \leftarrow 0$
2. If $\{t_d \mid (t_d \in TD) \wedge (\exists t_s \in TS : \neg\exists n \in N : (t_s,t_d,n) \in CD)\}$ is empty, go to step 7
3. $CD \leftarrow CD \bigcup \left\{ (t_s,t_m,l+1) \middle| (t_s,t_n,l) \in CD \wedge \begin{pmatrix} (\neg\exists\, n \in N : (t_s,t_m,n) \in CD) \wedge \\ IsSubCategoryOf(t_n,t_m) \wedge \\ HasLevel(t_n,L_n) \wedge \\ HasLevel(t_m,L_m) \wedge \\ L_n > L_m \end{pmatrix} \right\}$
4. If CD remains unchanged, go to step 7
5. $l \leftarrow l+1$
6. Go to step 2
7. $\forall((t_s,t_d,l) \in CD) \wedge (t_s \in TS) \wedge (t_d \in TD)$ return $Dist_{TT}^{SU}(t_s,t_d) = l$
```

Figure 3.19: Topic-to-topic strictly upward semantic distance calculator algorithm.

3.7.2.3 Topic-to-Topic Downward and Strictly Downward Semantic Distance

Although we will not be using downward distances here, it is very easy to define them based on their upward counterparts. Downward and strictly downward distances are the exact inverse of upward and strictly upward algorithms. Downward algorithm is specified by $Dist_{TT}^{D}(t_s,t_d)$ and can be defined by changing $IsSubCategoryOf(t_n,t_m)$ in step 3 of Figure 3.18 to $IsSuperCategoryOf(t_n,t_m)$. Similarly, strictly downward algorithm is specified by $Dist_{TT}^{SD}(t_s,t_d)$ and can be defined by changing $L_n > L_m$ and $IsSubCategoryOf(t_n,t_m)$ in step 3 of Figure 3.19 to $L_n < L_m$ and $IsSuperCategoryOf(t_n,t_m)$ respectively.

3.7.2.4 Entity-to-Topic Generic Semantic Distance

Calculating entity-to-topic semantic distance is trivial once the topic-to-topic semantic distance is defined. All we need to do is to find the topics of an entity, find the minimum distance from those topics to the target topic, and add the entity-to-topic edge constant weight (*c*) to the distance. To be generic the algorithm is presented for a set of initial entities (*IE*) and a set of target topics (*TT*). In this algorithm, topics of entity $e_i \in EI$ is referred to by $t_{i,j} \in TE$. Figure 3.20 shows the specification of the algorithm and presents the steps.

```
Goal:      Given a set of initial entities ($IE$) and a set of target
           topics ($TT$), find $Dist_{ET}(e_i,t_t)$ for any $e_i \in IE$ and $t_t \in TT$.
Input:     $IE \subset E$: Set of initial entities
           $TT \subset T$: Set of target topics
Output:    $Dist_{ET}(e_i,t_t)$ for all $e_i \in IE$ and $t_t \in TT$
1. $TE \leftarrow \left\{ t_{i,j} \mid (e_i \in IE) \wedge (t_{i,j} \in T) \wedge HasCategory(e_i, t_{i,j}) \right\}$
2. For each $e_i \in IE$, $t_t \in TT$, $t_{i,j} \in TE$, and $c \in R$, the distance
   $Dist_{ET}(e_i,t_t) = c + \min_j \left( Dist_{TT}(t_{i,j},t_t) \right)$
```

Figure 3.20: Entity-to-topic generic semantic distance calculator algorithm.

If entity-to-topic upward, strictly upward, downward, and strictly downward distances are needed to be calculated, it suffices to replace $Dist_{TT}(t_{i,j},t_t)$ in step 2 with the proper distance function from the corresponding topic-to-topic distances.

3.7.2.5 Entity-to-Entity Mixed Route Semantic Distance

As mentioned before, entity-to-entity distance can be calculated by either finding the minimum of the distances via different connection types or using a step-by-step exploration method. The former method is very similar to finding topic-to-topic or entity-to-topic generic semantic distance algorithms but with appropriate connections types. Here, we will discuss the latter method as it is the more complicated approach. In this algorithm, the distances between two sets of entities and all the intermediates are calculated. Figure 3.21 shows the specification of the algorithm and provides the steps.

In this algorithm, the distances between a starting set of entities (*ES*) and a destination set of entities (*ED*) and all the intermediate entities are calculated. The algorithm starts by giving a distance of zero from *ES* entities to themselves. It also defines *EN* as the set of entities to be processed next and is initialized to *ES*. Then, the algorithm starts exploring. Step 2 checks if any distances to *ED* are still outstanding, otherwise jumps to the end. Steps 3 to 6 respectively find the equivalent entities, the properties, the wikilinks, and entities with a common topic with *EN*. All of the new distance will be added unless a shorter distance to the new entities is already calculated. Then the newly added entities will replace the content of *EN* to be processed in next round. Step 8 will search within calculated distance and if there are two or more distances between the same two entities, it will keep the shortest one and discard the longer ones. The algorithm continues until all distances from *ES* to *ED* are calculated.

```
Goal:     Given a set of starting entities ($ES$) and a set of
          destination entities ($ED$), find $Dist_{EE}(e_s,e_d)$ for any
          $e_s \in ES$ and $e_d \in ED$.
Input:    $ES \subset E$ : Set of starting entities
          $ED \subset E$ : Set of destination entities
          $\alpha,\beta,\theta,c \in R$ : Weighting parameters
Output:   $Dist_{EE}(e_s,e_d)$ for all $e_s \in ES$ and $e_d \in ED$
1. $CD \leftarrow \{(e,e,0) \mid e \in ES\}$, $EN \leftarrow ES$
2. If $\{e_d \mid (e_d \in ED) \wedge (\exists e_s \in ES : \neg\exists x \in R : (e_s,e_d,x) \in CD)\} = \phi$, go to step 10
3. $CD \leftarrow CD \bigcup \left\{ (e_s,e_m,l) \;\middle|\; \begin{pmatrix} ((e_s,e_n,l) \in CD) \wedge \\ e_n \in EN \end{pmatrix} \wedge \begin{pmatrix} \begin{pmatrix} \neg\exists\, x \in R : ((e_s,e_m,x) \in CD) \wedge \\ (x \le l) \end{pmatrix} \wedge \\ \begin{pmatrix} RedirectsTo(e_n,e_m) \vee \\ DisambiguatesTo(e_n,e_m) \end{pmatrix} \end{pmatrix} \right\}$
4. $CD \leftarrow CD \bigcup \left\{ (e_s,e_m,l+\alpha) \;\middle|\; \begin{pmatrix} ((e_s,e_n,l) \in CD) \wedge \\ e_n \in EN \end{pmatrix} \wedge \begin{pmatrix} \begin{pmatrix} \neg\exists\, x \in R : \\ ((e_s,e_m,x) \in CD) \wedge \\ (x \le l+\alpha) \end{pmatrix} \wedge \\ \begin{pmatrix} HasProperty(e_n,e_m) \vee \\ HasProperty(e_m,e_n) \end{pmatrix} \end{pmatrix} \right\}$
```

```
5. $CD \leftarrow CD \bigcup \left\{ (e_s, e_m, l+\beta) \;\middle|\; \begin{pmatrix} ((e_s, e_n, l) \in CD) \wedge \\ e_n \in EN \end{pmatrix} \wedge \begin{pmatrix} \begin{pmatrix} \neg\exists\, x \in R : \\ ((e_s, e_m, x) \in CD) \wedge \\ (x \le l+\beta) \end{pmatrix} \wedge \\ \begin{pmatrix} LinksTo(e_n, e_m) \vee \\ LinksTo(e_m, e_n) \end{pmatrix} \end{pmatrix} \right\}$
6. $CD \leftarrow CD \bigcup \left\{ (e_s, e_m, l+2\cdot c\cdot\theta) \;\middle|\; \begin{pmatrix} ((e_s, e_n, l) \in CD) \wedge \\ e_n \in EN \end{pmatrix} \wedge \begin{pmatrix} \begin{pmatrix} \neg\exists\, x \in R : \\ ((e_s, e_m, x) \in CD) \wedge \\ (x \le l+2\cdot c\cdot\theta) \end{pmatrix} \wedge \\ \begin{pmatrix} \exists t \in T : \\ \begin{pmatrix} HasCategory(e_n, t) \wedge \\ HasCategory(e_m, t) \end{pmatrix} \end{pmatrix} \end{pmatrix} \right\}$
7. Replace $EN$ with the newly added entities
8. $CD \leftarrow CD - \{(e_s, e_n, l) \mid ((e_s, e_n, l) \in CD) \wedge (\exists (e_s, e_n, l') \in CD : l' < l)\}$
9. Go to step 2
10.   $\forall ((e_s, e_d, l) \in CD) \wedge (e_s \in ES) \wedge (e_d \in ED)$ return $Dist_{EE}(e_s, e_d) = l$
```

Figure 3.21: Entity-to-entity mixed route semantic distance calculator algorithm.

The above algorithm concludes how to define the semantic distance and how it is calculated in different configurations. While semantic distance is the main foundation of semantic analysis, it is not the only factor. The next step is to discuss how the semantic distance is played out versus a concept called commonality which represents how to handle multiple routes from entry points of the semantic graph to the target entities or topics.

3.7.3 Semantic Distance and Commonality

As described in the entity detection algorithm, every input component produces multiple candidate entities as the entry points to the semantic graph. Given the entry points, the problem is usually finding entities or topics that are similar to the input. Regardless of whether the target entities/topics are specified or not, the semantic distance is not the only deciding parameter in finding the most semantically similar target. That is because of multiple entry points will cause multiple routes to the targets and, therefore, there might be a trade-off between targets with closest semantic distance and targets being pointed by multiple entry points. In other words, a target might not be closest individual to any of the entry points but it might be the closest common target among most of the entry point. This factor is called the *commonality* of the target.

To clarify consider the following example. Suppose that input component has 10 candidate entities. Also suppose each entity has 3 topics on average. Now suppose there are 4 target topics to which we want to classify the input. To classify input to one of these 4 targets, we need to consider all possible routes from input entity topics to the target topics. Therefore, the number of topics resulted from entry point entities are about $3\times10 = 30$ topics considering 3 topic per candidate entity. This means the total number of possible routes to target topics is $30\times4 = 120$. Now, let us assume that candidate entity #4 has a topic which has a semantic distance of 3 to target topic #1 and this is the shortest distance among all other 120 possible routes. The point of commonality is that just because topic #1 has the shortest semantic distance to one of the input candidate entities, it is not necessarily the best topic to classify the input to. The classifier needs to find the topic which is commonly closer to all of the input candidate entities. In this example, the classifier can use an aggregate function, such as sum or average (depending on the problem), over all the possible routes to each target (30 for each target here) and calculate the overall semantic distance for each target topic.

Another aspect of commonality factor to consider is the score of each entity candidate. When detecting entities, each candidate received a separate score. This means there are different confidence levels for various entry points. Thus, the targets should be ranked based on the entry point scores as well as the number of routes and semantic distance of each route. How to define the effects of the candidate entity scores, the aggregation function applied on various routes, and the semantic distances on the final target scores depends on the application of the analysis. However, we can have some general rules of thumbs:

1. The higher candidate entity scores produce higher target scores.
2. The more routes to a common target probably[5] contribute to higher target scores.
3. The shorter semantic distances produce higher target scores.

[5] This might not always be true as it depends on the aggregation function e.g. average function can produce higher scores for fewer routes.

$$\begin{pmatrix} (M(e_c) \in X) \wedge ((X = E) \vee (X = T)) \wedge \\ (y_g \in Y) \wedge ((Y = E) \vee (Y = T)) \wedge \\ (AGG \in \{SUM, AVG, MAX, MIN, \cdots\}) \end{pmatrix} \tag{3.49}$$

$$\Rightarrow S_g^* \propto \underset{c}{AGG}\left(\frac{f(S^n(e_c))}{h(Dist_{XY}(M(e_c), y_g))} \right)$$

Equation (3.49) formalizes above rules in a generic way. In this equation, S_g^* on the left hand side (of the consequent) represents the target (goal) score for an entity or topic indexed *g*. On the right hand side, there is an aggregate function (*AGG*) over all the possible routes from all the candidate entities (e_c) to the target (y_g). Some examples of the aggregate function are given in the antecedent. Inside the aggregate function, there is fraction in which the numerator represents rule 1 above by making target score directly proportional to candidate entities scores. Note that the function (*f*) applied on the candidate entities should not render the proportion contradict rule 1 (unless otherwise intended). The denominator, on the other hand, follows rule 3 by making target score inversely proportional to the distance between candidate entities and the target. Again, note that the functions applied on the semantic distance (*h*) should support the inverse proportionality (unless otherwise intended). The *f* and *h* functions are scalar functions provided for flexibility on the power of the effect of nominator and denominator and can be customized in different applications. The function *M* on the other hand is an optional mapping function between candidate entities and other entities/topics. For instance if the distance is calculated between entities (i.e. *X* and *Y* are both equal to *E*), function *M* can be an identity function returning e_c as is. However, if the application is topic identification, *M* would be the *HasCategory* function which finds the topics of the candidate entity and the goal (y_g) would also be the target topics which the query should be classified in.

3.7.4 Entity Analysis

Entity analysis is applied on the types of problems which only involve entities. These kinds of problems start with a set of initial entities. These initial entities are either given or detected by the entity detection algorithm. Afterwards, there would be a number of alternating entity detections and semantic analyses until the input is fully processed. In these kinds of problems, there is often a set of conditions that the input entities should conform to. The entity detection algorithm finds the candidate entities conforming to the given conditions and scores them. Then, the semantic analysis component re-scores the candidate entities based on their similarity to the initial set of entities. The similarity is defined based on semantic distance and commonality of entities. Once the new candidate entities are re-scored and selected, the initial set of entities is updated based on the old and new entities and the process repeats. Some actual applications of entity analysis, such as speech recognition and text prediction, are given in Section 3.8. Below a generic template for entity analysis algorithm is described in Figure 3.22.

```
Goal:     Given a textual input source, find entities conforming to
          a set of conditions ($COND$).
Input:    $COMP_n$: Input components
          $COND$: A set of conditions entities should conform to
Output:   Result entities $RE_n$ for each component $COMP_n$ which
          conform to $COND$
1. $OE \leftarrow \{\ \}, \quad i \leftarrow 1$
2. Find candidate entities $E_i$ using the entity detection algorithm
   over $COMP_i$ and keep scores in $S^i(e_c)$
3. Filter out entities from $E_i$ which do not conform to $COND$
4. Find semantic distance between $OE$ and $E_i$
5. Calculate commonality score $S_c^*$ for all candidate entities $e_c \in E_i$
   using observed entities in $OE$, re-score all $E_i$ using $S^i$ and $S_c^*$
6. Select high scored entities from $E_i$ and put them in result
   entities $RE_i$ and return $RE_i$ as a part of output
7. Update $OE$ based on $RE_i$ : $OE \leftarrow u(OE, RE_i)$
8. $i \leftarrow i+1$
9. If $COMP_i$ exists, go to step 2, otherwise exit
```

Figure 3.22: Generic entity analysis template algorithm.

Note that in above algorithm, *OE* is empty in the first round; therefore, steps 4 and 5 are ignored in the first iteration and step 6 only uses the entity detection score ($S^1(e_c)$) to select result entities (RE_1). Afterwards, *OE* will be updated using result entities. The updating function (u) can be a simple replacement of *OE* by RE_i or it can mix all the entities and select the top ranked ones. The progressive analysis mentioned later will demonstrate another strategy for updating based on the age of an entity as well as its score. Finally, rescoring can be done by a customized function over S^i and S_c^*.

3.7.5 Topic Analysis

Topic analysis is applied on types of problems which require the input to be mapped to a certain set of topics. Therefore, the first step is to use the entity detection algorithm to provide the candidate entities for each input component (if text breaking is needed). Afterwards, the topics of candidate entities will be acquired. Finally, the distances between topics of candidate entities and the target topics will be calculated and scored using commonality equation. The target topic(s) with highest scores will be deemed as the topic of the input component. Topic identification application in Section 3.8 and most of Chapter 4 is dedicated to address the details topic analysis. One thing to note here is the possibility of having static or dynamic target topics in topic analysis. If the target topics are known and limited beforehand (static), it is possible to pre-calculate and cache the distances between target topics and all other topics in the topic graph. This action will increase the speed of algorithm considerably. However, if the targets are defined dynamically, the distances must be calculated in an iterative manner explained earlier. To increase the efficiency of the algorithm in this case, a limit can be imposed on the steps of iterative distance calculation and if the target topics are not within a threshold radius of starting topics, they will be considered as having the maximum possible distance or infinity.

3.7.6 Progressive Analysis

Semantic analysis in some problems requires the system to be constantly and dynamically involved with input. In other words, as the input components are read by the system, the semantic analysis is performed on each component and the results is kept and evolved for the next incoming input components. For instance, in a question-answering dialog system, the user keeps asking for new information and there is a good chance that the new questions are related to the past question. This system should keep track of the context of conversation and reason based on the history of the conversation. Although the natural language dialog system is not the scope of this research, the methods described earlier can assist this system to detect correct entities and topics.

Here, progressive analysis is applied in two directions. One over entity analysis which is called the context awareness application and the other is over topic analysis which is called topic tracking application both explained in Section 3.8. The progressive entity analysis tries to keep record of the entities already seen. In order to achieve this, the entities which win over thresholds i.e. score high enough to be selected as result entities will be stored in a set of observed entities (*OE*) similar to *OE* in the generic entity analysis algorithm. These observed entities will be assigned an age initialized to zero. Age of observed entities is incremented every iteration of semantic analysis. If an observed entity is observed again during next iterations of the semantic analysis, the age of that entity is reset to zero. In each round of semantic analysis, entities older than a threshold (T_{age}) are discarded from *OE* list. This mechanism keeps an updated memory of entities and these entities can, in turn, help to detect more correct entities from the input. Figure 3.23 summarizes the aging mechanism in progressive entity analysis. In this mechanism OE_i are the observed entities in the current iteration similar to RE_i in generic entity analysis algorithm.

```
Input:    $OE$ : General observed entities set
          $OE_i$ : The observed entities of current iteration($i$)
1. For all $\{e \mid ((e, age) \in OE) \wedge (e \in OE_i)\}$, do $OE \leftarrow OE - \{(e, age)\} \cup \{(e,0)\}$
2. For all $\{e \mid ((e, age) \notin OE) \wedge (e \in OE_i)\}$, do $OE \leftarrow OE \cup \{(e,0)\}$
3. If $((e, age) \in OE) \wedge (age > T_{age})$ then $OE \leftarrow OE - \{(e, age)\}$
```

Figure 3.23: Progressive analysis aging mechanism.

Applying progressive analysis on topics is similar to entities with the difference that the observed entities (*OE*) set is replaced with observed topics (*OT*). In each round of analysis, the target topics are selected by the analysis algorithm and they will be added to the *OT* set. If the new topics already exist in the *OT*, their age will be reset to zero. In each round, all the topics older than a threshold will be discarded. Both lists, *OE* and *OT*, are used to score the new candidate entities or topics in the new rounds of analysis. The rationale behind such a progressive analysis mechanism is that the entities in or topics of the input are coherently connected in successive components of the input. However, the change in context can happen but it is gradual. That is why the sets will discard the entities or topics which are not updated in the context after a while.

3.8 Applications

This section is dedicated to the introduction of a number of applications which can benefit from the entity detection and semantic analysis algorithms discussed earlier. The goal is to introduce a variety of different problems with different input and output types and for each problem, discuss how every element of semantic analysis is customized to address the issues of that specific problem. In each case, a potential solution is proposed. However, the proposed solution is not the only possible approach to solve that problem. It is possible to tackle these problems with tools other than the ones mentioned in this research. It is also possible to tackle these problems using the semantic analysis mentioned earlier but with various configurations. The potential solutions here provide one (possibly most generic) configuration. For one specific application (topic identification) various configurations are discussed in Chapter 4. The starting application is called Parla project and it is the initial motivation behind this research.

3.8.1 Parla Project

The Parla project was defined as a project with multiple research aspects. In general, it was a search engine for media content. The goal was to design a system which can provide a portal for accessing audio and video material using search abilities which can search through the content of an audio or video file similar to how the regular search engines search within textual web pages. This system would need a number of components to work properly. The main components of such a system will be:

- Automatic speech recognition engine to convert audio/video to text
- Search and indexing engine for information retrieval purposes
- User interface as a portal for users to access and work with the system

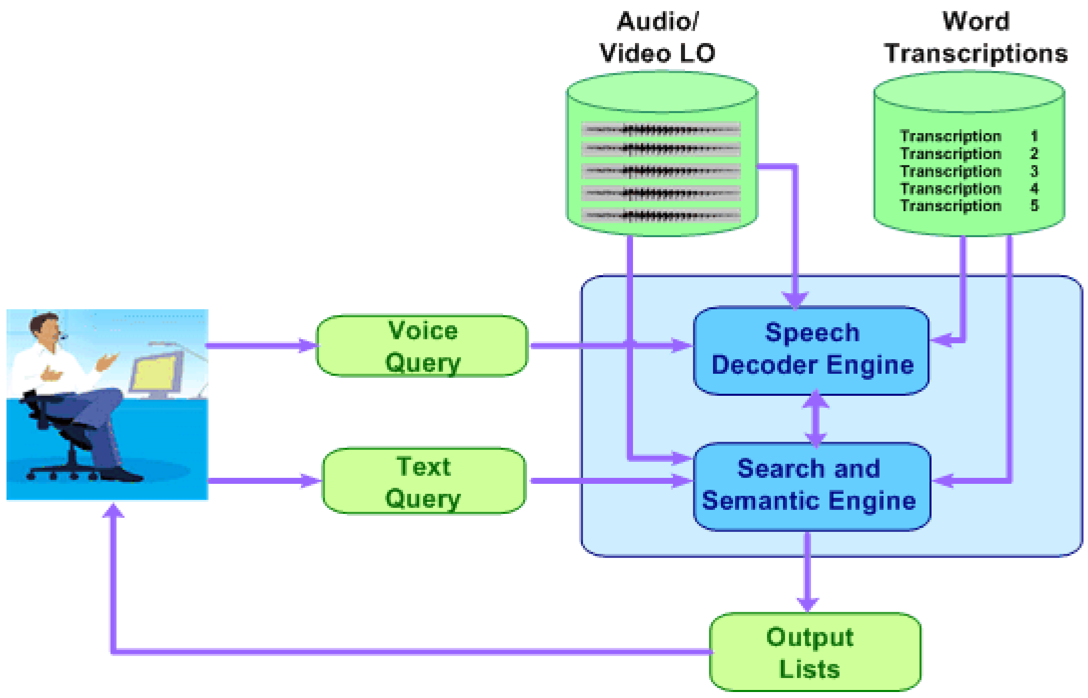


Figure 3.24: The structure of Parla project.

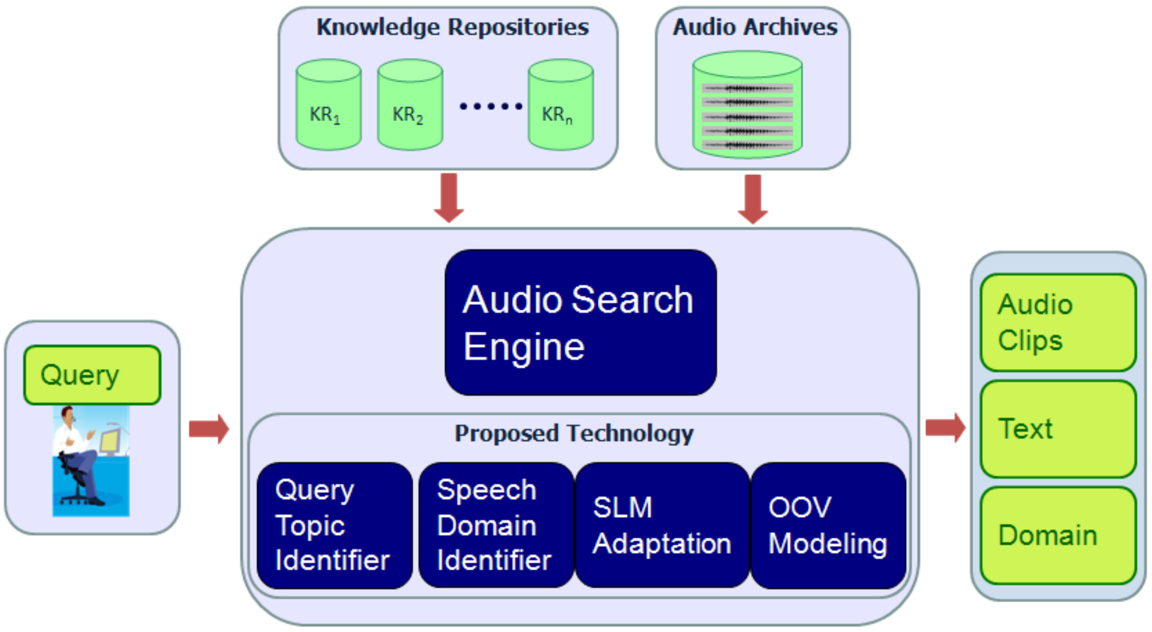


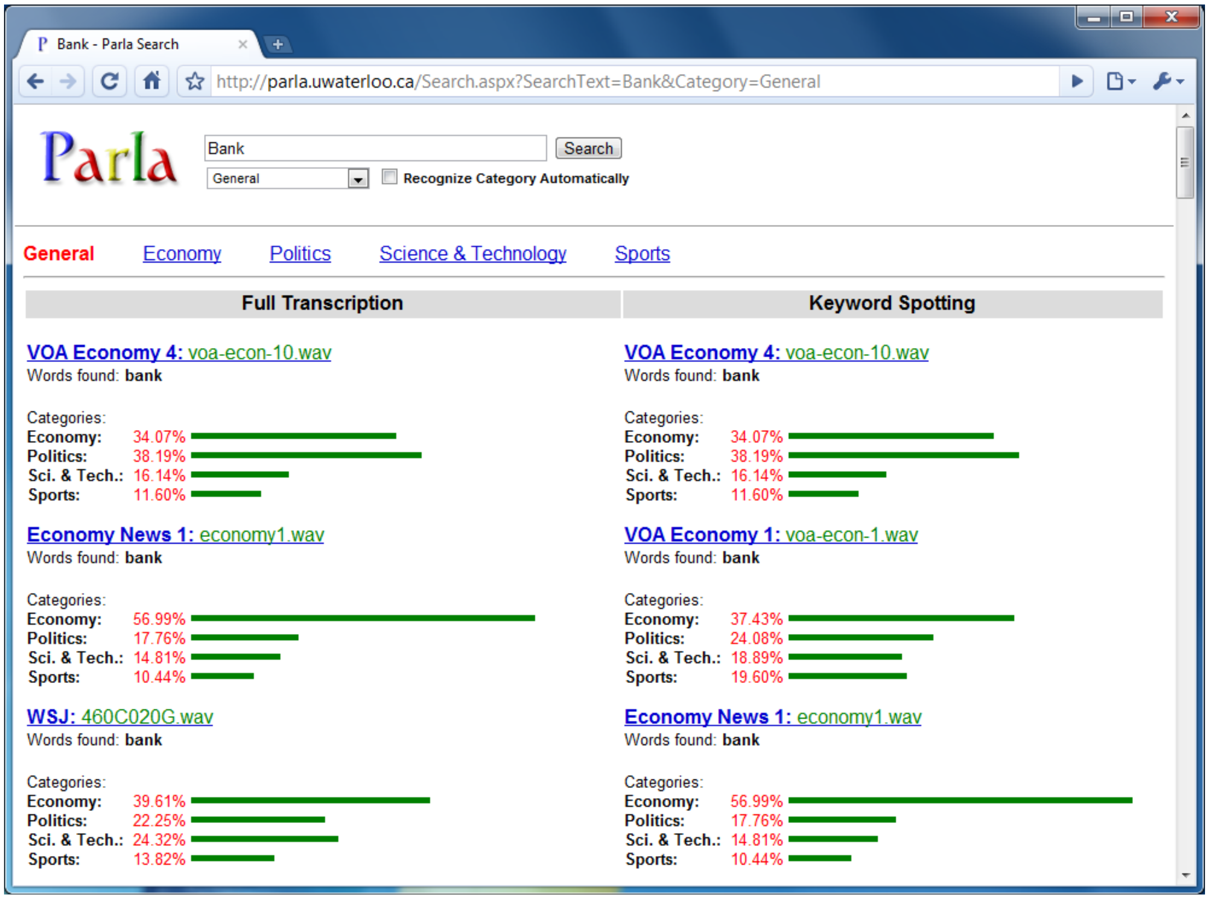
Figure 3.25: The components of the Parla project.

However, a generic speech recognition engine cannot perform reliably for all possible inputs from all possible topics. To increase the performance of the speech recognition engine, a semantic analysis engine was also included in the system. The semantic engine task was to perform two actions. First, after a transcription of an audio file was acquired using the generic speech recognition engine, the semantic engine would classify the text into a topic. Once the topic of the audio file was known, it was processed again using a speech recognition engine with a model dedicated to the topic of the audio file with higher accuracy. The second action performed by the semantic engine was to identify the topic of the user's query to the system and provide results related to the user's query topic. This last part of semantic engine which we call "Query Topic Identification" was the motivation behind this research. Figure 3.24 and Figure 3.25 show the final structure and components of the Parla project.

The *query topic identifier* component of the Parla project shown in Figure 3.25, is the component of interest here. The other components of the project are not discussed as they are out of the scope of this dissertation. The Parla project, however, is a nice showcase of how the semantic analysis methods explained earlier can be used in a real-world problem. The Parla project was limited to four topics for practical and developmental reasons but the concept works for as many desired topics. Therefore, the query topic identification task was to receive a user text query and identify to which of the four specified topics it belongs. The four topics used in the project were: Economy, Politics, Science & Technology, and Sports. Figure 3.26 shows a web page of the final system in action displaying the search results for the term 'bank'. To present the solution used to develop the query topic identifier component, the algorithm should be formally described as in Table 3.7.

Table 3.7: Query topic identifier algorithm description.

Algorithm:	**Parla Component: Query Topic Identifier**
Goal:	Given user's query, determine to which of four topics it belong.
Input:	*W*: Input query as bag of words (no necessary order)
Output:	t_i: One of the four topics (t_1, t_2, t_3, t_4)


Bank - Parla Search
http://parla.uwaterloo.ca/Search.aspx?SearchText=Bank&Category=General
Parla
Search
Recognize Category Automatically
General
Economy
Politics
Science & Technology
Sports

Bank
General

Full Transcription

VOA Economy 4: voa-econ-10.wav
Words found: **bank**

Categories:
Economy: 34.07%
Politics: 38.19%
Sci. & Tech.: 16.14%
Sports: 11.60%

Economy News 1: economy1.wav
Words found: **bank**

Categories:
Economy: 56.99%
Politics: 17.76%
Sci. & Tech.: 14.81%
Sports: 10.44%

WSJ: 460C020G.wav
Words found: **bank**

Categories:
Economy: 39.61%
Politics: 22.25%
Sci. & Tech.: 24.32%
Sports: 13.82%

Keyword Spotting

VOA Economy 4: voa-econ-10.wav
Words found: **bank**

Categories:
Economy: 34.07%
Politics: 38.19%
Sci. & Tech.: 16.14%
Sports: 11.60%

VOA Economy 1: voa-econ-1.wav
Words found: **bank**

Categories:
Economy: 37.43%
Politics: 24.08%
Sci. & Tech.: 18.89%
Sports: 19.60%

Economy News 1: economy1.wav
Words found: **bank**

Categories:
Economy: 56.99%
Politics: 17.76%
Sci. & Tech.: 14.81%
Sports: 10.44%

Figure 3.26: A screenshot of the Parla system in action.

The next step is to define each part of solution according to the methodology described in Section 3.6 and Section 3.7. Therefore, the first action is to define input for the entity detection algorithm. Since a textual query submitted to a search engine is not necessarily an ordered text, the best model to describe the query is a bag of words as defined in Equation (3.17). Moreover, it is assumed that the query is, in nature, a short text containing a few words [100]. This means that the text breaking is not necessary and the input would be only one component. Furthermore, stop words are not removed in Parla project. Therefore, the input would be $Comp_1 = \{k_1, k_2, \cdots, k_N\}$ in which N is the total number of input words and k_i are the input words i.e. keywords.

Once the search division of the entity detection algorithm finds the candidate entities with one or more input keywords, a score for candidate entities should be defined. The score for candidate entities are based on Equation (3.25) with no ordering weight and character length as the penalty weight.

Therefore, the score for candidate entities is equal to $S^1(e_c) = N_k \times C_k / C_e$. Afterwards, the equivalency division adds redirects with the score mentioned but the disambiguations were not added as they were supposed to be scored as separate entities based on their annotated main titles e.g. “Phoenix (mythology)” for “Phoenix”. The production division would also apply a limit of top 100 on the entity detection output. However, the limit is applied on topics of candidate entities rather than candidate entities themselves. The reason is that in order to map the input to one of the four target topics, the extracted candidate entities should be mapped to their topics and then the semantic analysis can be performed between the topics of the candidate entities and the target topics. The topics of candidate entities are called base topics and shown as t_b which means for a certain candidate entity $HasCategory(e_c, t_b)$ is true i.e. t_b is a topic of e_c. Once all the base topics are gathered, a score parameter can be defined for each base topic. First, a parameter called topic-to-keyword weight $W_{k_i}^{t_b}(e_c)$ is defined as:

$$\begin{pmatrix} Contains\left(e_c, k_i\right) \wedge \\ HasCategory\left(e_c, t_b\right) \end{pmatrix} \Rightarrow W_{k_i}^{t_b}(e_c) = S^1(e_c) \tag{3.50}$$

Topic-to-keyword weight specifies that if the candidate entity e_c has the keyword k_i and is related to the topic t_b, then the $W_{k_i}^{t_b}(e_c)$ weight is equal to the score of that candidate entity. Once all the possible topic-to-keyword weights are calculated, a parameter called *density* is defined which basically represents the score of base topic. The density of each base topic t_b is defined as:

$$D(t_b) = \sum_{k_i} \max_{e_c}\left(W_{k_i}^{t_b}(e_c)\right) \tag{3.51}$$

Once the density of all base topics are calculated, the production division ranks the base topics based on their densities and keeps the top 100 ones and discards the rest.

The final part of the solution is to specify what type of semantic distance is used and what parameters are used to define the commonality score of target topics. The semantic distance used in this project is the strictly upward topic-to-topic distance between base topics and the target topics because the four target topics are very general topics and in fact most of them have a level 1 standing in the topic graph. The strictly upward condition was chosen rather than upward to avoid multiple routes in distance calculation. Equation (3.52) expresses the commonality Equation of (3.49) customized for the query topic identifier task of the Parla project. As you can see, the Density function here, acts as the *f* function of the original commonality equation since it converts the score of candidate entities to a new value. The mapping function (*M*) is the *HasCategory* which asserts that t_b is a topic of the candidate entity. Finally, the *h* function here is the cubic power of the distance plus a small value. The small value is added to avoid division-by-zero in case of zero distances. The *h* function is empirically tuned for this specific application and has no theoretical rationale.

$$g \in \{1,2,3,4\} \Rightarrow S_g^* = \sum_b \left(\frac{D(t_b)}{\left(Dist_{TT}^{SU}(t_b, t_g)\right)^3 + 0.00001} \right) \tag{3.52}$$

In Equation (3.52), t_g indicates one of the four the target topics and S_g^* indicates the final score of each of the target topics. Once all four scores are calculated, the target topic with the highest score is selected as the topic of the input query and the results of the search engine shown in Figure 3.26 are filtered out accordingly. It is notable that in the development of the query topic identifier component of Parla project, the Wikipedia categories used to represent the four target topics are in fact 5 categories. “Science” and “Technology” are each an individual category. Therefore, to tackle the problem, the semantic analysis is in fact performed using 5 target topics but those two topics are considered as one when the highest scored topic is returned. This application showcased the use of semantic analysis in a real-world problem. The next application reiterates the steps of topic identification application explained here with the difference of being a generic problem rather than the specific case discussed here.

3.8.2 Topic Identification

Topic identification is the main application used in this research. While the setup of the application is similar to the Parla project explained above, the topic identification is defined as the generalization of the Parla project. While, in this section, the general approach is described, Chapter 4 provides a comprehensive study of the parameters involved and the affects of each of those parameters. To define topic identification problem, we can use a generalization of query topic identifier component which states: given a textual query, find the top topics from a set of specified topics which identify the query best semantically. Thus, the difference with Parla project is two-fold. One, the target topics can be any number; two, the result can be one or more target topics. This means the topic identification should be able to match a query to any number of target topics. Figure 3.27 provides the formal description of the topic identification algorithm and iterates the steps of a potential generic solution.

```
Goal:     Given user's query, find top $M$ topics semantically most
          relevant to the query.
Input:    $W$: Input query as a bag of words (no necessary order)
          $T_G$: The set of target topics
Output:   $(t_g, r_g)$: A subset of $T_G$ indicating semantically most
          relevant topics to the query and their ranks ($r_g$)for
          $(t_g \in T_G) \wedge (1 \le g \le M) \wedge (M \le \|T_G\|)$
1. Pre-process the input by removing stop words and creating one
   component: $Comp_1 = \{K_1, \varnothing, \varnothing\}, K_1 = \{k_i | (1 \le i \le N) \wedge (\exists w \in W : k_i = w)\}$
2. Find all the candidate entities $(e_c, INFO_c) \in E_1$ using $Comp_1$
3. Calculate the score of all candidate entities: $S^1(e_c) = N_k \times N_k / N_e$
4. Find all base topics: topics of all candidate entities ($t_b$)
5. Calculate all topic-to-keyword weights:
   $Contains(e_c, k_i) \wedge HasCategory(e_c, t_b) \Rightarrow W_{k_i}^{t_b}(e_c) = S^1(e_c)$
6. Calculate density of all base topics: $D(t_b) = \sum_{k_i} \max_{e_c}(W_{k_i}^{t_b}(e_c))$
7. Calculate score of all target topics: $S_g^* = \sum_b \left( \frac{D(t_b)}{e^{Dist_{TT}(t_b, t_g)}} \right)$
8. Sort all $t_g$ based on $S_g^*$ descending, rank $t_g$ based on its position
   as $r_g$, and return top M as $(t_g, r_g)$
```

Figure 3.27: Topic identification algorithm.

The above algorithm shows a similar approach to the Parla project query topic identifier algorithm mentioned earlier. However, there are a few differences which are applied on the generic approach to fit most cases rather than the specific case of Parla project. Below is a list of differences and their justifications:

- The stop words are removed generally as is common practice. Based on the application and whether the query includes the exact search quotation marks, removing the stop words can be applied or ignored on different parts of the input query.
- The candidate score penalty weight uses the number of featured words to the number of total entity words here. This is a generic choice. If an application regards longer words as more important words, the character count proportion might be a better choice.
- The base topics are not limited in this generic approach. All the possible base topics will affect the final score of target topics.
- The chosen semantic distance is generic topic-to-topic distance as we do not know in advance if the target topics are general ones or specific ones. Therefore, no upward or downward specification should be applied here.
- The *h* function applied on the distance function is an exponential function. The reason for choosing this function is that it gives higher weight to closer topics than the further ones. Due to the nature of exponential function, higher distances will produce very large denominators which, in turn, shrink the effect of that specific topic in the summation severely. Therefore, the closer base topics would be the dominant ones in shaping the final score of a target topic. There are, however, many other possibilities for the *h* function as discussed in Chapter 4. But the exponential function is shown a good generic function especially when the base topics are not limited (compare 25 to 100 base topics in Section 4.2).

3.8.3 Speech Recognition

The application of semantic analysis in speech recognition is a very specific one for a specific setup. The semantic analysis can be applied as a secondary algorithm over the speech recognition algorithm in order to improve the performance of the speech recognition. Here the setup is briefly explained and the potential solution is described. Later in Section 4.3, the details of the algorithm are discussed.

In certain speech recognition methods, after the audio data of a sentence is processed, it is possible that two or more candidate sentences are produced as the result of the recognition algorithm. Each of these sentences might include right or wrong transcribed words. For instance, it is possible that the first two words of the first sentence are the correct transcriptions while words 3, 4, and 5 of the second sentence are the correct ones. Therefore, there should be an algorithm which finds all possible combinations of the words and decides which word from which sentence should be selected as the final transcription result of the speech recognition algorithm. Moreover, this process will be repeated for the next recognized sentences of the audio file. While simple voting algorithm might be used in individual sentence processing, a smart semantic analysis can help improve the performance if there is more than one sentence in the audio file and the sentences are semantically connected as any regular conversation is.

The input for this application consists of sets of equal length sentences. Each set has equal number of sentences compared to other sets but the number of the words in each sentence of each set can be different from other sets. For example, first set has 3 sentences each with 10 words. The second set has 3 sentences with 6 words each and so on. It is possible that there are blank words in sentences to make up for the missing words. The text breaking algorithm in this case does not do much as the input is already in a structured format. Assuming that each set of sentences has *I* sentences and there are *M* total sets of sentences, $w_{i,j}^{m}$ indicates the *j*th word of the *i*th sentence of *m*th set in which $1 \le i \le I$ and $1 \le m \le M$. The number of words in each set is different. Thus, the number of words in the set *m* is specified by N_m .

The desired output of this application is a sequence of words for each set of sentences which creates one final sentence created out of the words of *I* sentences in each set. The order of words is important here. This means the *jth* word of the final sentence for set *m*, should be the *jth* word from one of the *I* sentences in set *m*. Therefore the results can be shown as an ordered sequence of words each from a sentence *i* of each set: $Output_m = \left\langle w_{idx_1^m,1}^{m}, w_{idx_2^m,2}^{m}, \cdots, w_{idx_j^m,j}^{m}, \cdots, w_{idx_{N_m}^m,N_m}^{m} \right\rangle$ in which $idx_j^m \in \{1,2,\cdots,I\}$. For instance, for *I* = 3, *m* = 2, and N_2 = 4, the output can be: $Output_2 = \left\langle w_{2,1}^{2}, w_{1,2}^{2}, w_{3,3}^{2}, w_{1,4}^{2} \right\rangle$ which means the selected words are: first word from second sentence, second word from first sentence, third word from third sentence, and fourth word from first sentence of set 2.

Solving this problem can be performed at two stages. At the first stage, the entities of each set can be found individually and only based on how well they fit the possible combinations. Based on the candidate entities, the correct combination of words can be suggested. At the second stage, the entities of earlier sets can be used as clues to determine the entities of next sets more accurately. Therefore, the second stage is a progressive entity analysis. Here, a generic two stage solution is provided while in Section 4.3, a more detailed stage one solution is described along with an example. Figure 3.28 illustrates the specification and the algorithm steps for the potential solution.

```
Goal:     Given a number of sets of multiple sentences, find the
          best combination of words creating one sentence for each
          set.
Input:    $M$ :   Number of sets
          $I$ :   Number of sentences in each set
          $N_m$ : Number of words in set $m$
          $w^m_{i,j}$ : $j^{th}$ word of $i^{th}$ sentence of the set $m$
Output:   $Output_m = \left\langle w^m_{idx^m_1,1}, w^m_{idx^m_2,2}, \cdots, w^m_{idx^m_j,j}, \cdots, w^m_{idx^m_{N_m},N_m} \right\rangle$ for $1 \le m \le M$

1. $m \leftarrow 1, OE \leftarrow \{\ \}$
2. Find all candidate entities matching any exact sequence of $w^m_{i,j}$
   and score them as $S^m(e_c) = N_k \times N_k / N_e$
3. If $OE \neq \{\ \}$, find the semantic distance between $OE$ and candidate
   entities, find commonality score $S^*_c$ for all candidate entities,
   re-score candidate entities based on $S^m$ and $S^*_c$
4. Sort candidate entities descending over their scores
5. Select the top candidate as the winning entity, select the words
   $w^m_{i,j}$ featured in the winning candidate as the final results in
   $Output_m$, add the winning entity to: $OE_m$
```

```
6. If all word slots in the current set are decided go to step 9
7. Based on the selected words in step 5, re-score candidate
   entities considering possible conflicts, remove the winning
   entity and all entities which do not provide words for remaining
   slots from the candidate entity list
8. Go to step 4
9. Perform progressive analysis aging mechanism using $OE$ and $OE_m$
10.  $m \leftarrow m+1$
11.  If $m > M$ return $Output_m$ for all sets, otherwise go to step 2
```

Figure 3.28: Speech recognition word combination solver algorithm.

In the above algorithm, only steps 3 and 9 are related to the second stage which is progressive entity analysis. One can remove these steps and the algorithm still works on an individual level as we will observe in Section 4.3. The points of importance in above algorithm can be summarized as following:

- The entity detection scoring equations uses word number penalty weight as the length of words are not important in this application. Moreover, since the search of words is an exact match to a sequence of the input words for that set, the words are found in order. This means that N_k is equal to O_k in this case. We will see a better alternative scoring for this application which considers how many sentences are suggesting the same word for a certain position and that becomes the weight of that word. Instead of having N_k, the sum of the weight of words can now be used as the word featuring score along with penalty weight.
- After selecting one winning entity and using it to specify words for certain positions, the already found candidate entities can now support combinations that have conflicting words in the position which algorithm already decided for. These conflicts should be considered and that is why a recalculation of both N_k and penalty weight should be performed and the entities should be re-scored (step 7).

- The general observed entities (*OE*) act as the context tracker (as we will see in last application of this section). It means that the candidate entities which are closer to observed entities (which get updated every iteration) are probably better candidates. Therefore, after finding the entity-to-entity semantic score S_c^*, the original score of the candidate entities S^m can be updated accordingly. The simple approach for this updating is to normalize S_c^* to [0, 1] and then use $S^m(e_c) = (1 + S_c^*) \times S^m(e_c)$ to boost the scores. The entity with highest S_c^* will have its S^m score doubled and the one with lowest S_c^* will have its S^m untouched.
- The winning entities of each set constitute the observed entities of that iteration (OE_m) using which the aging mechanism will update the general *OE*. This means the *OE* will be regularly updated and it follows the context of the audio data.

3.8.4 Text Prediction

Text prediction is another potential problem on which the semantic analysis can be applied. There are many commercial methods in text prediction especially in mobile application development. The best example is SwiftKey keyboard application [104] for Android devices. This product uses a sophisticated probabilistic approach for predicting what the next word the user wants to type would be and suggests that. Suggesting words facilitates typing on touch devices as the keyboards are not physical and, in case of mobile phones, are very small which make typing harder than usual keyboards. The product is now moving towards providing context dependent keyboards as well. For instance, it provides SwiftKey healthcare version which is a customized version for people using their devices in a healthcare environment. This version considers healthcare topics to be the dominant ones when predicting words and also contains vocabulary and knowledgebases dedicated to healthcare, such as medicine and illness names, which are not that common in everyday use. The solution provided here follows a similar rationale of trying to guess the context of the text and then suggest words (or phrases) based on the context acquired using previous words read from the input. Therefore, the approach is a bit different in a way that the suggestions in this solution are dynamically determined based on the conversation context while products such as SwiftKey have pre-calculated *n*-gram probability tables.

The problem definition here is as follows: given a continuous stream of words as input, provide a list of possible words when a few starting letters of the current word are given. This means the prediction happens after a number of starting letters is known and, therefore, the list of possible words are limited to the ones which start with those known letters. However, those letters can still point to many possible words which means the algorithm should have a method for ranking possible words and return the top *N* ones as the answer. Note that the number of predicted words (*N*) is usually limited depending on the screen size of the phone or other user interface issues. Therefore, it is important that the selection of top *N* is performed accurately. Moreover, the number of letters given before a prediction is made (designated by *M* here) can be variable too[6]. Figure 3.29 provides the algorithm specification and the steps.

```
Goal:      Given a continuous stream of words as input, provide a
           list of possible words when a few starting letters of the
           current word are given.
Input:     $M$ :   Number of letters of the current word received
                   before performing prediction
           $N$ :   Number of predicted words to return
           $P$ :   Number of previous words to include in $n$-gram
                   entity detection
           $I$ :   Current word position
           $L_I$ : Number Starting letters of the current word ( $w_I$ )
           $w_i$ : Previous input words for $1 \le i \le (I-1)$ and current word
                   for $i = I$
Output:    $Output_I = \langle pw_1, pw_2, \cdots, pw_N \rangle$ are $N$ predicted words for
           current position ( $I$ )
1. $i \leftarrow 1, OE \leftarrow \{ \}$
2. Read next $M$ letters from input, if no more entry to input, exit
3. Find all candidate entities which start with "$L_I$" or "$w_{I-1}\ L_I$" or
```

[6] $M = 3$ is a common choice.

```
   "$w_{I-2}\ w_{I-1}\ L_I$" or until "$w_{I-P} \cdots w_{I-2}\ w_{I-1}\ L_I$"; score the candidate
   entities using $S^I(e_c) = N_k \times N_k / N_e$
4. If $OE \neq \{\ \}$, find the semantic distance between $OE$ and candidate
   entities, find commonality score $S_c^*$ for all candidate entities,
   re-score candidate entities based on $S^I$ and $S_c^*$
5. Sort candidate entities descending over their scores
6. Return $N$ distinct words $Output_I = \langle pw_1, pw_2, \cdots, pw_N \rangle$ that start with
   $L_I$ letters from the corresponding top $N$ candidate entities
7. Observe the complete current word $w_I$, if it matches any of top N
   predicted words, add the candidate entity containing that word
   into $OE$, update $OE$ using aging mechanism
8. $i \leftarrow i+1$
9. Go to step 2
```

Figure 3.29: Text prediction algorithm.

The potential solution provided above is a progressive entity analysis algorithm which attempts to keep a list of observed entities and rank candidate entities based on the context created by the observed entities. Furthermore, the algorithm not only tries to find candidate entities starting with the known letters but also entities which might match previous words plus the known words. For instance, suppose that the observed input so far is: “The constitution of the united” and the next 3 letters of the next word are: “sta”. The algorithm here tries to find all the candidate entities which start with letters “sta” such as “Stable”, “Standard”, and “State” as well as entities which match previous words plus the next word known letters such as “United States” which matches the previous word of the input “united” and then the 3 letters “sta” in word “States”. The number of previous words considered in this fashion is another parameter of the algorithm which is denoted by *P* here. Finally, the observed entity list is updated using the aging mechanism. However, the age parameter used here should be larger than the one used, for example, in speech recognition application. The reason is that in an application like speech recognition, the update happens after many entities were recognized in an input set and added to the observed entity list. This simulates a batch update. In this application, however, the update happens after every single word prediction. Therefore, the entities can become outdated very quickly which is not desirable and thus, the age threshold (T_{age}) should be a larger number.

The above algorithm is yet another example of progressive entity analysis used to select better candidate entity at every pass of the algorithm over the input. There are a few points about the above algorithm which can be discussed further as following:

- The candidate entities are matched to the current word letters (L_I) and P previous words using exact match and, therefore, the words are all in order. However, each candidate entity might include extra words after the word starting with read letters. For instance, continuing the example mentioned earlier, the entity "United States of America" matches the input "The constitution of the united sta…" but has two extra words "of America". The preference is to find an entity which has lesser extra words and match the input completely hence the scoring method of $S^I(e_c) = N_k \times N_k / N_e$.
- When returning predicted words, the algorithm goes through the candidate entities and returns only the word that matches the L_I instead of the entire entity. For instance, in case of "United States of America" being the candidate entity, the word "States" is returned rather than all the rest of the entity "States of America". In some variations, the developer might decide to actually suggest multiple consecutive words in each entry. In that case, the other words of the entity can and will be used.
- Since the user will eventually enter the full word, the algorithm can use this feedback and tune itself for further predictions. In this algorithm, such a tuning is performed by adding candidate entities of correctly predicted words into the observed entity list and creating a context for the input seen up to that point.
- Note that in the updating of *OE*, there is no current loop observed entity list (OE_i). In other words, the OE_i is only consisted of the correctly predicted candidate entities. Therefore the aging mechanism only updates the correctly predicted candidate entities in each round.

3.8.5 Context Awareness and Topic Tracking

The last application discussed here are, in fact, more of meta-applications meaning that they are tools used in other applications to enhance their performance. We have already observed the context awareness application as the progressive entity analysis used in both speech recognition and text prediction applications. Keeping track of the context of the input can be useful in most of text processing solutions. Keeping track of the topic of the input can provide a second layer of semantic awareness for the different algorithms. In this application, the way the context and topic of the input are used is not discussed. This application only provides the method to recognize the context and the topic of the input. The context and topic can then be used to perform a variety of actions including finding the semantic distance of new candidate entities to the recognized context entities or topics and re-scoring them. As an example, both the context and topic can be used to perform word sense disambiguation, a common task in most natural language processing and semantic processing problems.

The problem definition here is simply to firstly, keep track of the input context which we define as a set of observed entities and secondly, keep track of the topic(s) of the input. The topic(s) will be selected from a pre-specified set of topics. It can be one or more topics returned for each component of the input. Both the context and the topics are recognized based on candidate entities in each round of the algorithm. The candidate entities are extracted from input as before but it is possible that extra conditions of the system affect the candidate entity list. These extra conditions are called (*COND*) in the algorithm.

To tackle this problem, two lists are kept: observed entities (*OE*) and observed topics (*OT*). The observed entities are selected from top scored candidate entities of each round (similar to two previous applications). The observed topics are recognized using generic topic identification algorithm over the candidate entities of each round. Both lists are updated using aging mechanism. The age threshold can, however, be different for each list. Figure 3.30 provides the description of the algorithm as a pseudocode.

As shown in Figure 3.30, this algorithm does not provide specific results. It only demonstrates how to create and maintain the *OE* and *OT* lists. Step 12 provides the opportunity to add extra steps for applications that use context and topic hence making this algorithm a customizable template.

```
Goal:      Given a stream of input words, track context and topic of
           the input.
Input:     $w_i$:       Input word at position $i$
           $T_G$:       A set of goal topics for tracking the topic
           $N$:         Number of topics to return each round
           $ST$:        Candidate entity score threshold
           $COND$:      Specific problem conditions which candidate
                        entities should follow
Output:    $OE$ and $OT$

Input: $w_i, T_G, N, ST, COND$

1.  $OE \leftarrow \{\ \}, OT \leftarrow \{\ \}, i \leftarrow 1$
2.  Perform text breaking method appropriate for the application
    and create $COMP_i$, if no more input available, exit
3.  Find candidate entities $E_i$ using the entity detection
    algorithm over $COMP_i$ and keep scores in $S^i(e_c)$
4.  Filter out entities from $E_i$ which do not conform to $COND$
5.  If $OE \neq \{\ \}$, find semantic distance between $OE$ and $E_i$
6.  Calculate commonality score $S_c^*$ for all candidate entities
    $e_c \in E_i$ using observed entities in $OE$, re-score all $E_i$ using
    $S^i$ and $S_c^*$
7.  Select high scored entities from $E_i$ with score over $ST$, put
    them in $OE_i$
8.  Use $E_i$ and find top $N$ goal topics ($t_n \in T_G$) for $1 \leq n \leq N$ using
    generic topic identification algorithm
```

```
9.   Put top $N$ goal topics in $OT_i$
10.  Update $OE$ based on $OE_i$ using aging mechanism
11.  Update $OT$ based on $OT_i$ using aging mechanism
12.  If needed, use $OE$ and $OT$ to enhance the system performance
13.  $i \leftarrow i+1$
14.  Go to step 2
```

Figure 3.30: Context awareness and topic tracking algorithm.

3.9 Summary

This chapter introduced and explained the proposed methodology of this dissertation in great details. First, it was explained how to create a knowledgebase using Wikipedia or DBpedia. These two sources were comprehensively studied and they resulted in the final knowledgebase called the semantic graph. Next the components of the semantic graph were described. These components were a topic graph over the entity graph linked by entity-topic connection layer. Once the concepts of entity and topic were sufficiently explained, the main algorithms of the methodology were introduced. First, the entity detection algorithm was discussed and formalized. Then, the semantic analysis algorithms were presented. Semantic distance, which is the main component of semantic analysis, was defined between entities and topics and the algorithms for its calculation were expressed. Afterwards, the second most important component of semantic analysis which is commonality score was presented. Based on these two components, various semantic analysis namely entity analysis, topic analysis, and progressive analysis were introduced and explained. Finally, a list of real-world applications was provided and a potential solution for each of these applications was offered. The solutions were studied and important points were explained. The next chapter will study further two of the applications provided above.

Chapter 4
Experimental Results

4.1 Introduction

In this chapter, a comprehensive study and the related experimental results of studying various effects of the parameters in the two applications introduced in Section 3.8 is presented. Query topic identification is the main application on which the semantic analysis algorithms are applied in this study. There are many parameters in the algorithms which can be studied for perfect performance. This chapter main focus is to study the effect of these parameters and show the trends associated with the variations of each parameter. This chapter would also provide a vision into the speech recognition application. It provides an entity detection algorithm along with its results and compares them with the results of a baseline method.

4.2 Query Topic Identification

As mentioned in the previous chapter, query topic identification was a component of the Parla project which started this research into the applications of semantic analysis using an encyclopedic knowledgebase. In order to achieve the goals of both the Parla project and this research, numerous experiments were designed and performed so to study various parameters and their effects. Moreover, these experiments helped shaping the methodology explained in the previous chapter. In other words, the described methodology is the result of incremental empirical testing as well as theoretical studies. Therefore, the first two experiments would not match the described methodology completely but it shows how the building blocks of the methodology are shaped as well as how other approaches compare to the described methodology. The upcoming sections summarize these efforts into three experiments.

The first experiment shows the very early attempts in learning whether using the entity detection algorithm can withstand the noise words and if so, how resistant the entity detection algorithm is towards various amount of noise words added to the queries. At this stage, there is no semantic analysis in form of semantic distance or commonality scoring. The presented algorithm only uses the density function introduced in Section 3.8. The data used in this toy problem are all synthesized using Wikipedia articles.

The second experiment provides results for an iterative algorithm which uses an upward exploration strategy in finding parent topics until it reaches the target topics. This method presented in this experiment has unique characteristics. Firstly, the result of the algorithm can be a variable number of target topics since it follows a step-by-step exploration method and it is possible that in each step a different number of target topics are met. Secondly, there is no explicit semantic distance in this method as the target topics are selected using a ‘first visited, first selected’ approach. This means implicitly the topics with smaller semantic distance are selected but there is no commonality applied.

Finally, the third experiment is the main experiment which is exactly according to the semantic analysis methodology. There are various setups in this experiment showing the effects of various parameters, such as the number of base topics and the *h* function in commonality equation. These last two experiments are both using the KDD CUP 2005 datasets.

4.2.1 Toy Problem: Synthesized Data

In this experiment, a query is synthesized using a random entity and then it is polluted with unrelated words called noise words. The aim of the experiment is to study how resistant the entity detection algorithm is to the noise i.e. how much noise words can be added without losing considerable performance accuracy. To describe the setup, three aspects should be explained.

4.2.1.1 Data Synthesis

To create the input data for this experiment, a number of queries are generated. Each query is associated with a topic and, therefore, each query has a number of words or phrases which are taken from entities related to that topic. Each query has also a number of noise words which are random words unrelated to the query topic. The following simple steps are performed to generate a single query associated with its topic:

1. Select a random topic and save it as the target topic of the query
2. Select a random entity which belongs to the selected topic and add it to the query. Repeat this step until the desired numbers of keywords (or key-phrases) are added to the query.
3. Select and add random unrelated words to the query until the desired ratio between keywords of step 2 and the noise words is acquired.

4.2.1.2 Identification Algorithm

The algorithm used to identify the topic of each query is similar to the entity detection algorithm explained for the Parla project and generic topic identification application in Section 3.8. In a nutshell, given the query, the algorithm finds all the candidate entities containing at least one word of the query. It only uses the number of featured words as candidate entity score (no penalty or ordering weights). Then, the keyword-to-topic weights are calculated. Based on these weights, the densities of the topics are acquired. The top 100 topics are then sorted descending over their densities. Figure 4.1 depicts the steps of the algorithm.

```
Input: The query as a bag of words
1. Pre-process the input by removing stop words and creating one
   component: $Comp_1 = \{K_1, \varnothing, \varnothing\}, K_1 = \{k_i \mid (1 \le i \le N) \wedge (\exists w \in W : k_i = w)\}$
2. Find all the candidate entities $(e_c, INFO_c) \in E_1$ using $Comp_1$
3. Score candidate entities using $S^1(e_c) = N_k$
4. Find all base topics: topics of all candidate entities ($t_b$)
5. Calculate all topic-to-keyword weights:
   $Contains(e_c, k_i) \wedge HasCategory(e_c, t_b) \Rightarrow W_{k_i}^{t_b}(e_c) = S^1(e_c)$
6. Calculate density of all base topics: $D(t_b) = \sum_{k_i} \max_{e_c} \left( W_{k_i}^{t_b}(e_c) \right)$
7. Sort all $t_b$ over $D(t_b)$ descending
8. Keep the top 100 base topics and discard the rest
```

Figure 4.1: Toy problem topic identification algorithm.

4.2.1.3 Evaluation Metrics

The common metrics used in evaluation experiments in text classification tasks are the popular *recall*, *precision*, and *F1* measures. However, due to randomly synthesized nature of the data in this experiment, we only know to which topic a query belongs but we do not know to which other topics the query does or does not belong. Therefore, when classifying a query, we can discern whether it is classified in correct topic (true positive) or it is not classified in correct topic (false negative) which means we can only calculate recall which is the ratio of true positives to the sum of true positives and false negatives. To specify whether a query is classified correctly or not, the topic of the query is searched in the final top 100 topics returned by the algorithm above. If the target topic has the first rank in the list, the query is classified correctly.

A second metric is also used in this experiment. While the recall metric would tell us what percentage of the queries were classified correctly, it does not provide any extra information about the queries not classified correctly. It is desired to know if the topic of a query is not identified as the first choice, where in the list the topic is located. In other words, it is desired to know the relative location of target topics in the returned list so even if a query is not classified correctly, we would be able to see if it was classified close to top of the list rather down below. This second metric is called *percentile*. It, however, differs slightly from the regular percentile definition. A percentile here indicated 100% minus the percentage of the topics with a higher score. e.g. if 5 items have the same highest score in a list containing 100 different items, the percentile for all of those top 5 items is 100% (rather than 95%). The 6th item which has slightly lesser score will have a percentile of 95%.

To have a better evaluation in the following setups, each setup is repeated 100 times and the recall and percentile metrics were averaged over all repetitions to provide more precise results. The following setups test three factors. The first setup tests the effect of the absolute number of noise word given a constant number of keywords. The second setup studies the query size versus the ratio of noise words. Last but not least, the third setup represents the effect of having an ordered search (exact search) versus no order of words in the query. The semantic graph used in the three setups is the Wikipedia database dump of September 2008 which contains about 5.5 million entities (titles) and about 400,000 topics (categories) before clean-up and about 260,000 after clean-up and leveling.

4.2.1.4 Setup 1: Noise Words Effect

In the first setup, we try to make a preliminary observation on the effect of noise words on our method. The input consists of one key phrase from one entity and a variable number of noise words. The number of noise words varies from 0 to 20. Figure 4.2 shows the results for both measures. The red solid line with diamond markers shows the recall metric and the blue dashed line with circle markers indicates the percentile metric. Note that the recall metric is originally in range [0, 1]. Here it is mapped to [0, 100] in order to be depicted with the percentile metric in the same scale.

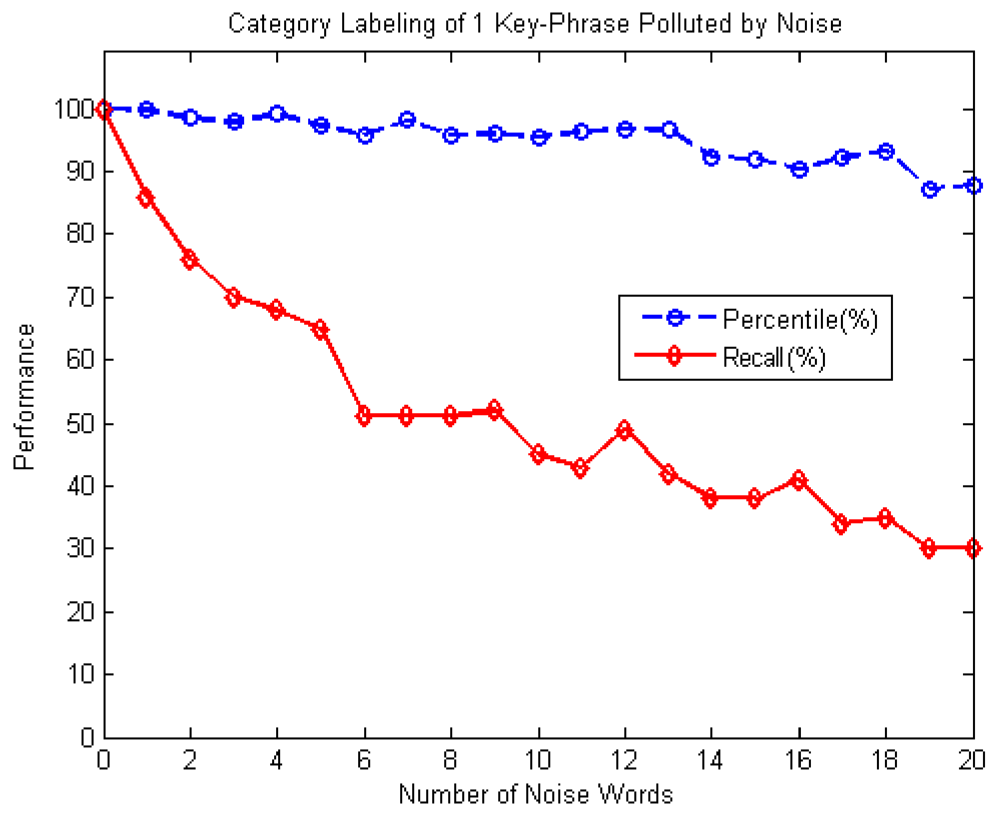


Figure 4.2: Toy problem setup 1 performance chart.

Since there is only one key phrase in the experiment, it is expected that the addition of noise words affect the performance considerably. The probability of noise words forming a false entity increases when the number of noise words increases, specifically when it surpasses the number of the keywords of the key phrase. Therefore, the decline of recall is justified. Percentile, on the other hand, is consistently high even with the presence of excessive noise words. This means that the target topic is always among the top returned topics even when there is high amount of noise words present.

4.2.1.5 Setup 2: Query Size versus Noise Ratio

In the second setup, we are trying to examine how much the ratio of the key phrases in a query to the number of noise words affects the performance. The input consists of a number of key phrases and noise phrases. In this experiment, we change the ratio of noise while keeping the total number of phrases constant in order to cover a wide range of combinations. The total number of phrases would be 10, 20, 30, and 40 and we apply noise with the ratio of 0% to 90%. For example, when the total number of phrases is 20 and the noise ratio is 40%, we would have 12 key phrases and 8 noise phrases. Figure 4.3 and Figure 4.4 show the results for recall and percentile respectively.

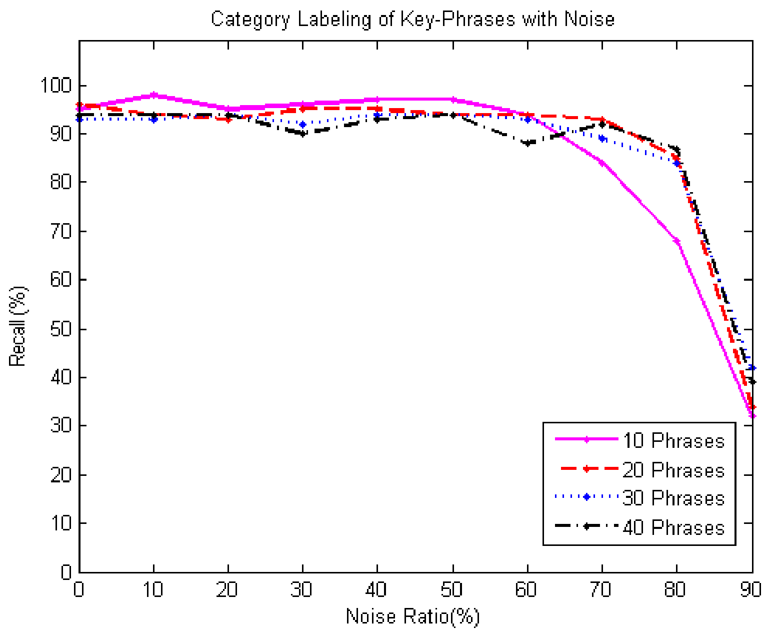


Figure 4.3: Toy problem setup 2 recall performance chart.

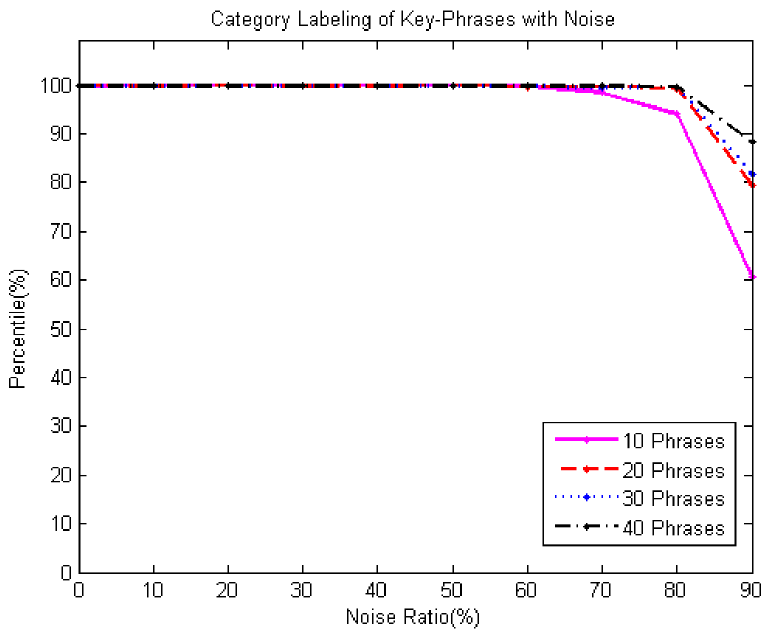


Figure 4.4: Toy problem setup 2 percentile performance chart.

It is evident from the results that the method can guaranty more than 99% percentile and 90% recall performance when the noise ratio is less than 70%. In other words, the method can effectively resist the noise up to 70% without a major drop in performance. Such a resistance threshold is high enough for our purpose considering the noise ratio will not usually be too high. Such a high performance is specifically significant if we consider the size of the goal set which is a set of 260,155 different topics after some clean-up.

4.2.1.6 Setup 3: Effect of Ordered Query

The third setup is identical to the second setup except for the query which is defined as exact phrases rather than phrases i.e. query is keeping the order of words as in using quotation marks around the key-phrases taken from entities. The results in Figure 4.5 and Figure 4.6 clearly show that both metrics improve strongly when the query is ordered. We can observe about 10% increase in noise resistance and more similarity between recall and percentile evaluations compared to having no order. The charts suggest that the queries with exact search quotation marks might have up to 80% noise words and still keep a high performance in base topic identification. This setup concludes the toy problem experiment. The next experiment will incorporate real query data and comparisons with real baseline methods.

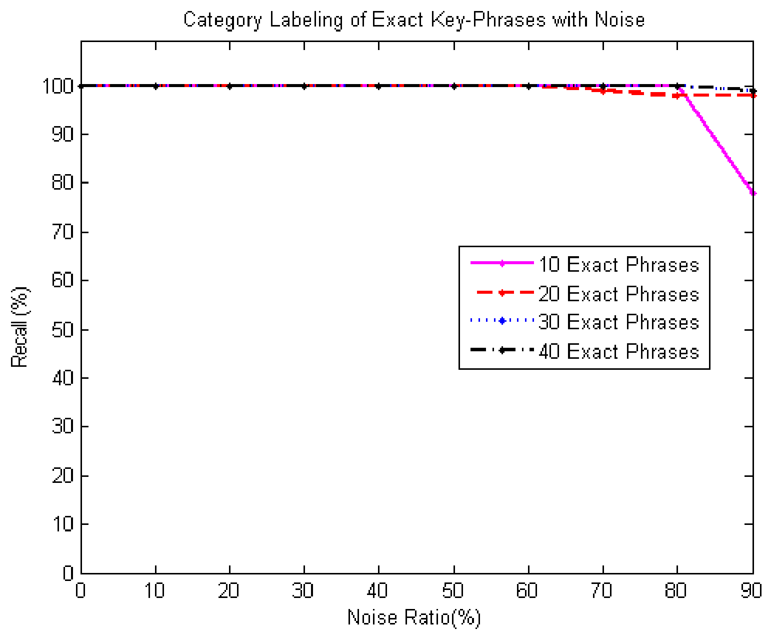


Figure 4.5: Toy problem setup 3 recall performance chart.

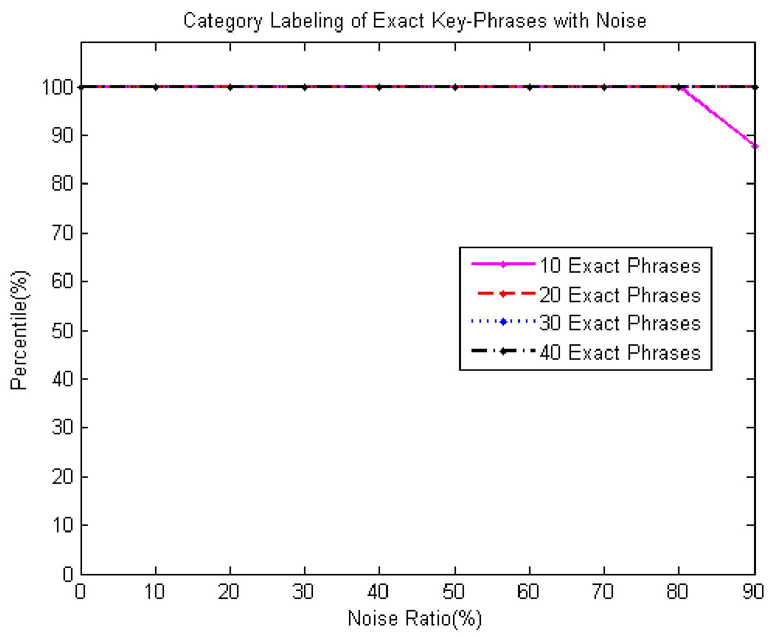


Figure 4.6: Toy problem setup 3 percentile performance chart.

4.2.2 Iterative Algorithm

The query classifier proposed in this experiment is designed to explore the topic graph from any starting point until it reaches the nearest specified target topics. The main characteristic of this algorithm is that it can return multiple topics per query and it is the user's choice to specify the maximum number of topics a query can belong to (specified by *ClassificationSize* in the algorithm). The algorithm will then return topics up to that number as the result. The pseudocode of this algorithm is shown in Figure 4.7. For this experiment, the semantic graph is again created from the Wikipedia version available from September 2008.

```
Define:    TargetTopics (a subset of the topic graph),
           Classification (classification results),
           CassificationSize (number of classification results
           allowed per query)
Input:     Query

1. Classification ← { }

2. EntityList ← the candidate entities after stop word removal
```

```
3. TopicList ← the base topics of the candidate entities
4. Do for 20 iterations:
        a. NewClassification ← subset of TopicList that are in
           TargetTopics
        b. If COUNT(Classification + NewClassification) <=
           CassificationSize
             i. Classification ← Classification +
                NewClassification
        c. If COUNT(Classification + NewClassification) >
           CassificationSize AND COUNT(Classification) > 0
             i. Break from loop
        d. If COUNT(Classification + NewClassification) >
           CassificationSize AND COUNT(Classification) = 0
             i. Classification ← Select CassificationSize elements
                from NewClassification
            ii. Break from loop
        e. TopicList ← unvisited parent topics directly connected
           to TopicList
    5. Return Classification
```

Figure 4.7: Iterative query topic identification algorithm.

4.2.2.1 Exploration Algorithm

The first step of this algorithm is to map the user's query to an initial set of base topics from which the exploration of the graph will begin. This is accomplished by the entity detection algorithm as before. The query is stripped of stop words to keep only keywords; the system then generates the exhaustive list of candidate entities that feature at least one of these keywords, and expands the exhaustive list of base topics pointed to by these candidate entities. Next, the algorithm considers each keyword/entity/topic triplet where it is the case that the keyword is in the entity and the entity points to the topic, and assigns each one the keyword-to-topic weight that is equal to word featuring weight and character penalty weight equal to $W_{k_i}^{t_b}(e_c) = S^1(e_c) = N_k \times C_k / C_e$. As a reminder, N_k is the total number of query keywords featured in the candidate entity, C_k is the character count of the keywords featured in the candidate entity, and C_e is the total number of characters in the candidate entity. The rationale for using character counts in this experiment is to shift some density weight to entities that match longer keywords in the query. The assumption is that, given that the user typically only provides less than four keywords in the query, having one much longer keyword in the set could mean that this one keyword is more important. Consequently, we give a higher weight to keywords in an entity featuring the longer query keywords and missing the shorter ones, as opposed to an entity featuring the shorter query keywords and missing the longer ones. Finally, the density value of each base topic is computed again using: $D(t_b) = \sum_{k_i} \max_{e_c}\left(W_{k_i}^{t_b}(e_c)\right)$.

This process will generate a long list of topics, featuring some topics pointed to by high-weight keywords and summing to a high density score, and a lot of topics pointed to by only lower-weight keywords and having a lower score. The list is trimmed by discarding all topics having a score less than half that of the highest-density topic (as explained by variable score threshold in production division of the entity detection algorithm). This trimmed set of topics is the initial set the exploration algorithm will proceed from. It corresponds to "TopicList" at step 3 of the pseudocode in Figure 4.7. Through practical experiments, we found that this set typically contains approximately 28 topics.

Once the initial list of topics is available, the search algorithm explores the topic graph step by step. At each step, the algorithm compares the set of newly-visited topics to the list of target topics defined as acceptable classification labels and adds any targets discovered to the list of classification results. It then generates the next generation of unvisited topics directly connected to the current set as parent and repeats the process. The exploration can thus be seen as radiating through the graph from each initial topic. This process corresponds to steps (a) and (e) of the algorithm.

There are two basic termination conditions for the exploration algorithm. The first is when a predefined maximum number of classification results have been discovered. This maximum could for example be 1, if the user wants a unique classification for each query, while it was set at 5 in the KDD CUP 2005 competition rules. However, since the exploration algorithm can discover several target topics in a single iteration, it is possible to overshoot this maximum. The algorithm has two possible behaviors defined in that case. First, if some results have already been discovered, then the new topics are all rejected. For example, if the algorithm has already discovered four target topics to a given query out of a maximum of five and two more topics are discovered in the next iteration, both new topics are rejected and only four results are returned. The second behavior is for the special case where no target topics have been discovered yet and more than the allowed maximum are discovered at once. In that case, the algorithm simply selects randomly the maximum allowed number of results from the set. For example, if the algorithm discovers six target topics at once in an iteration, five of them will be kept at random and returned as the classification result.

The second termination condition for the algorithm is reaching a maximum of 20 iterations. The rationale for this is that, at each iteration, both the set of topics visited and the set of newly-generated topics expand. The limit of 20 iterations thus reflects a practical consideration, to prevent the size of the search from growing without constraint. But moreover, after 20 steps, we find that the algorithm has explored too far from the initial topics for the targets encountered to still be relevant. For comparison, in our experiments, the exploration algorithm discovered the maximum number of target topics in only 3 iterations on average, and never reached the 20 iterations limit. This limit thus also allows the algorithm to cut off the exploration of a region of the graph that is very far removed from target topics and will not generate relevant results.

4.2.2.2 Experimental Results

In order to test this algorithm, it was submitted it to the same challenge as the KDD CUP 2005 competition [53]. The 37 solutions entered in that competition were evaluated by classifying a set of 800 queries into up to 5 categories (topics) from a predefined set of 67 target categories c_i, and comparing the results to the classification done by three human labelers. The 800 text queries were meaningful English queries selected randomly from MSN search logs, unedited and including the users' typos and mistakes. The solutions were ranked based on overall precision and overall F1 value, as computed by equations (4.1) to (4.6). The competition's Performance Award was given to the system with the top overall F1 value, and the Precision Award was given to the system with the top overall precision value within the top 10 systems evaluated on overall F1 value. Note that participants had the option to enter their system for precision ranking but not F1 ranking or vice-versa rather than both precision and F1 ranking, and several participants chose to use that option. Consequently, the top 10 systems on F1 value ranked for precision are not the same as the top 10 systems ranked for F1 value, and there are some N/A values in the results in Table 4.1.

$$Precision = \frac{\sum_i \texttt{Number of queries correctly labeled as } c_i}{\sum_i \texttt{Number of queries labeled as } c_i} \quad (4.1)$$

$$Recall = \frac{\sum_i \texttt{Number of queries correctly labeled as } c_i}{\sum_i \texttt{Number of queries belonging to } c_i} \quad (4.2)$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4.3)$$

$$Overall\ Precision = \frac{1}{3}\sum_{j=1}^{3} Precision \texttt{ against labeler } j \quad (4.4)$$

$$Overall\ Recall = \frac{1}{3}\sum_{j=1}^{3} Recall \texttt{ against labeler } j \quad (4.5)$$

$$Overall\ F1 = \frac{1}{3}\sum_{j=1}^{3} F1 \texttt{ against labeler } j \quad (4.6)$$

In order for this algorithm to compare to the KDD CUP competition results, we need to use the same set of category labels. Fortunately, the size and level of detail of Wikipedia's topic graph makes it possible to identify topics to map any set of labels to. In this case, we identified 84 target topics in Wikipedia corresponding to the 67 KDD CUP category set.

With the mapping done, the 800 test queries were classified using above algorithm and the results were evaluated on overall precision and F1 following the KDD CUP guidelines. Our results are presented in Table 4.1 along with the KDD CUP mean and median, the best system on precision, the best system on F1, and the worst system overall as reported in [53]. As can be seen from that table, our system performs well above the competition average, and in fact ranks in the top 10 of the competition.

Table 4.1: Iterative topic identification algorithm performance in baseline standings.

System	F1 Rank	Precision Rank	Overall Precision	Overall F1
Best F1	1	N/A	0.4141	0.4444
Best Precision	N/A	1	0.4237	0.4261
Our algorithm	10	7	0.3081	0.3005
Mean	18	13	0.2545	0.2353
Median	19	15	0.2446	0.2327
Worst	37	37	0.0509	0.0603

It is interesting to consider not only the final classification result, but also the performance of our exploration algorithm. To do this, we studied how frequently each of the termination conditions was reached. We can summarize that there are five distinct ways the algorithm can terminate. The first is "no initial list", which is to say that the initial keyword-to-topic mapping failed to generate any topics for our initial set and the exploration cannot begin. If there is an initial set of topics generated and the exploration begins, then there are still four ways it can terminate. The first is "failure", if it reaches the cutoff value of 20 iterations without encountering a single target topic. The second termination condition is "exploration limit", if the algorithm reaches the cutoff value of 20 iterations but did discover some target topics along the way. These topics are returned as the classification results. The third termination is the "overshoot", if the algorithm discovers more than the maximum number of results in a single iteration and must select results randomly. And the final termination condition is "topic limit", which is when the algorithm has already found some topics and discovers more topics that bring it to or above the set maximum; in the case it goes above the maximum, the newly-discovered topics are discarded. In each case, we obtained the number of query searches that ended in that condition, the average number of iterations it took the algorithm to reach that condition, the average number of topics found (which can be greater than the maximum allowed when more topics are found in the last iteration) and the average number of target topics returned. These results are presented in Table 4.2.

Table 4.2: Iterative topic identification algorithm exploration results chart.

Termination	**Number of Queries**	**Average Number of Iterations**	**Average number of target topics found**	**Average number of target topics returned**
No initial list	52	0	0	0
Failure	0	20	0	N/A
Exploration limit	0	20	0	N/A
Overshoot	28	2.4	7.0	5
Topic limit	720	3.3	7.8	3.3

As can be seen from above table, two of the five termination conditions we identified never occur at all. They are the two undesirable conditions where the exploration strays 20 iterations away from the initial topics. This result indicates that our exploration algorithm never does diverge into wrong directions or miss the target topics, nor does it end up exploring in regions without target topics.

However, there is still one undesirable condition that does occur, namely that of the algorithm selecting no initial topics to begin the search from. This occurs when no entities featuring query words can be found; typically because the query consists only of unusual terms and abbreviations. For example, one query consisting only of the abbreviation “AATFCU” failed for this reason. Fortunately, this does not happen frequently: only 6.5% of queries in our test set terminated for this reason. The most common termination conditions, accounting for 93.5% of query searches, are when the exploration successfully discovers the maximum number of target topics, either in several iterations or all in one, with the former case being much more common than the latter. In both cases, we can see that the system discovers these topics quickly, in less than 4 iterations on average. This shows the success and efficiency of our exploration algorithm. Moreover, the classification results compare favorably to those of the KDD CUP 2005 competition: this algorithm would have ranked 7th on precision in that competition, with an increase of 6.4% compared to the competition median, and 10th on F1 with a 6.9% increase compared to the median. The next experiment, however, tries to even improve these results further by using the complete semantic analysis methodology presented in this dissertation and tuning its parameters to create the optimal query identification system.

4.2.3 Semantic Distance Algorithm

This experiment is a natural continuation of the previous experiment. The semantic graph used is the same as last experiment with a slightly different cleaning method for the topic graph. There are 5,453,808 entities and 282,271 topics in this semantic graph. However, there are also a few fundamental differences. In this experiment, we intend to fully use the components of semantic analysis methodology. This means there is no step-by-step exploration and instead we are using the semantic distance method explained in the methodology. This causes that the number of returned topics for all queries be the same. Moreover, in contrast with previous experiment where we only allowed paths going from child to parent topic, we allow any path between two topics, regardless of whether it goes up to parent topics or down to children topics or zigzags through the graph. Therefore, the semantic distance is the generic $Dist_{TT}(t_b, t_g)$. The reason for adopting this more permissive approach is to make our classifier more general: the parent-only approach may work well in the case of previous experiment where all the goal topics selected were higher in the hierarchy than the average base topic, but it would fail miserably in the opposite scenario when the base topics are parents of the goal topics. Finally, we can note that, while exploring the graph to find the shortest distance from every goal topic and all other topics may seem like a daunting task, for a set of about

100 goal topics, such as we used in this experiment, it can be done in only a few minutes on a regular computer.

The identification algorithm used in this experiment is very similar to the generic topic identification algorithm described in Figure 3.27. It, however, has a few distinctions in order to make it possible to study the important aspects of the algorithm. Figure 4.8 shows the steps of this new algorithm. Note that the input to this algorithm is still the words of the query (*W*) and the set of target topics (T_G).

```
Input: $W, T_G$
1. Pre-process the input by removing stop words and creating one
   component: $Comp_1 = \{K_1, \varnothing, \varnothing\}, K_1 = \{k_i \mid (1 \le i \le N) \wedge (\exists w \in W : k_i = w)\}$
2. Find all the candidate entities $(e_c, INFO_c) \in E_1$ using $Comp_1$
3. Calculate the score of all candidate entities: $S^1(e_c) = N_k \times P_k$
4. Find all base topics: topics of all candidate entities ($t_b$)
5. Calculate all topic-to-keyword weights:
   $Contains(e_c, k_i) \wedge HasCategory(e_c, t_b) \Rightarrow W_{k_i}^{t_b}(e_c) = S^1(e_c)$
6. Calculate density of all base topics: $D(t_b) = \sum_{k_i} \max_{e_c} \left( W_{k_i}^{t_b}(e_c) \right)$
7. Keep the high density base topics and discard the rest
8. Calculate score of all target topics: $S_g^* = \sum_b \left( \frac{D(t_b)}{h\left(Dist_{TT}(t_b, t_g)\right)} \right)$
9. Sort all $t_g \in T_G$ based on $S_g^*$ descending and return top M
```

Figure 4.8: Query topic identification algorithm.

The first distinction of the algorithm is the score of candidate entities which is defined by a variable penalty weight. This is because the penalty weight is one of the parameters to be studied. In this experiment, all three penalty weights introduced by Equation (3.27) are studied. Regardless of the penalty weight, there is an interesting point about the keyword weights. We can see from the definition of $W_{k_i}^{t_b}(e_c)$ that every keyword appearing in an entity will receive the same weight $S^1(e_c)$. Moreover, when an entity is composed exclusively of query keywords (i.e. there is no penalty), their weight will be the number of keywords contained in the entity (N_k). The maximum weight a keyword can have is thus equal to the number of keywords in the query; it occurs in the case where an entity is composed of all query keywords and nothing else. In such case, we can see that the maximum density a topic can have is the square of the number of query keywords. It happens in the case where each keyword has its maximum value in that topic, meaning that one of the entities pointing to the topic is composed of exactly the query keywords.

Another distinction in the above algorithm is the addition of the step 7. At this stage of the algorithm, we have a weighted list of base topics, featuring some topics pointed to by high-weight keywords and summing to a high density score, and a lot of topics pointed to by only lower-weight keywords and having a lower score. In our experiments, we found that the set contains over 3,000 base topics on average. We limit the size of this list by keeping only the set of highest-density topics, as topics with a density too low are deemed to be too unrelated to the original query to be of use. This can be done either on a density basis (i.e. keeping topics whose density is more than a certain proportion of the highest density obtained for this query, regardless of the number of topics this represents, as we did in the previous experiment) or on a set-size basis (i.e. keeping a fixed number of topics regardless of their density, the approach we will prioritize in this experiment). When using the set-size approach, a question arises on how to deal with ties when the number of tied topics exceeds the size of the set to return. In this experiment, we break ties by keeping a count of the number of entities that feature keywords and that point to each topic, and giving priority to the topics pointed to by more entities.

The last distinction in the algorithm with generic topic identification algorithm is the *h* function which determines how much the effect of the semantic distance would be. In the algorithm, this function is left as variable so that different options can be studied. . There are of course some options that have been considered in the literature different than the semantic distance described in this methodology. For example, Coursey and Mihalcea [105] proposed an alternative metric based on graph centrality, while Syed *et al.* [96] developed a spreading activation scheme to discover related concepts in a set of documents. Some of these ideas could be adapted into our method in future research.

However, even after settling on the shortest-path distance metric, there are many ways we could take into account the base topics' densities into the goal topics' ranking. The simplest option is to use it at a threshold value – to cut off base topics that have a density lower than a certain value, and then rank the goal topics according to which are closest to any remaining base topic regardless of density. That is the approach we used in the previous experiment using exploration method. On the other hand, taking the density into account creates different conditions for the system. Since some base topics are now more important than others, it becomes acceptable, for example, to rank a goal that is further away from several high-density base topics higher than a goal that is closer to a low-density base topic. We thus define a ranking score for the goal topics, as the sum for all base topics of a ratio of their density to the distance separating the goal and base. There are several ways to compute this ratio; five options that we considered in this study are:

$$S_g^* = \sum_b \left(\frac{D(t_b)}{Dist_{TT}\left(t_b, t_g\right) + 0.0001} \right) \tag{4.7}$$

$$S_g^* = \sum_b \left(\frac{D(t_b)}{Dist_{TT}\left(t_b, t_g\right)^2 + 0.0001} \right) \tag{4.8}$$

$$S_g^* = \sum_b \left(\frac{D(t_b)}{e^{Dist_{TT}\left(t_b, t_g\right)}} \right) \tag{4.9}$$

$$S_g^* = \sum_b \left(\frac{D(t_b)}{e^{2 \times Dist_{TT}\left(t_b, t_g\right)}} \right) \tag{4.10}$$

$$S_g^* = \sum_b \left(\frac{D(t_b)}{e^{Dist_{TT}\left(t_b, t_g\right)^2}} \right) \tag{4.11}$$

In each of these equations, the score S_g^* of goal topic t_g is computed as the sum, for all base topics t_b, of the density $D(t_b)$ of that topic, divided by a function of the distance between topics t_b and t_g. This *h* function is a simple division in equations (4.7) and (4.8), but the exponential in equations (4.9), (4.10), and (4.11) put progressively more importance on the distance compared to the density. The addition of 0.0001 in equations (4.7) and (4.8) is simply to avoid a division by zero in the case where a selected base topic is also a goal topic.

Finally, the goal topics with the highest score are returned as classification results. In this experiment, we return the top three topics, to allow for queries to belong to several different topics. We believe that this corresponds to a human level of categorization; for example, in the KDD CUP 2005 competition [53], human labelers used on average 3.3 topics per query. However, this parameter is flexible, and we ran experiments keeping anywhere from one to five goal topics.

The various alternatives and options for our classifier described above were all implemented and tested, in order to study the behavior of the system and determine the optimal combination. That optimal combination was then subjected to a final set of tests with new data. In order to compare and study the variations of our system, we again submitted them all to the same challenge as the KDD CUP 2005 competition [53]. All the measures and metrics are similar to what was explained in previous experiment and equations (4.1) to (4.6). The only difference is that the mapping between KDD CUP goal category set and the topic graph were performed a little more comprehensively and we identified 99 goal topics in our topic graph corresponding to the 67 KDD CUP category set. These correspondences are presented in Appendix A.

4.2.3.1 Penalty Weight

The first aspect of the system we studied is the different formulae for the penalty weight of query keywords in an entity. The choice of formula has a direct impact on the system, as it determines which entities are more relevant given the user's query. This in turn determines the relevance of the base topics that lead to the goal topics. A bad choice at this stage can have an impact on the rest of the system.

The weight of an entity, and of the query keywords it contains, is function of the two parameters presented in Equation (3.25), namely the number of keywords present in the entity and the penalty for extra keywords in that entity. We have introduced three possible mathematical definitions of penalty weight for an entity. They are a straightforward proportion of keywords in the entity, the proportion of characters in the entity that belong to keywords, and the proportion of IDF of keywords to the total IDF of the entity, as computed with Equation (3.30). We implemented all three equations and tested the system independently using each. In all implementations, we limited the list of base topics to 25, weighted the goal topics using Equation (4.8), and varied the number of returned goal topics from 1 to 5. The results of these experiments are presented in Figure 4.9. The three different experiments are shown with different grey shades and markers: dark squares for the formula using the proportion of characters, medium triangles for the formula using the proportion of keywords, and light circles for the formula using the proportion of IDF. Three results are also shown for each experiment: the overall precision computed using Equation (4.4) in a dashed line, the overall recall of Equation (4.5) in a dotted line, and the overall F1 of Equation (4.6) in a solid line.

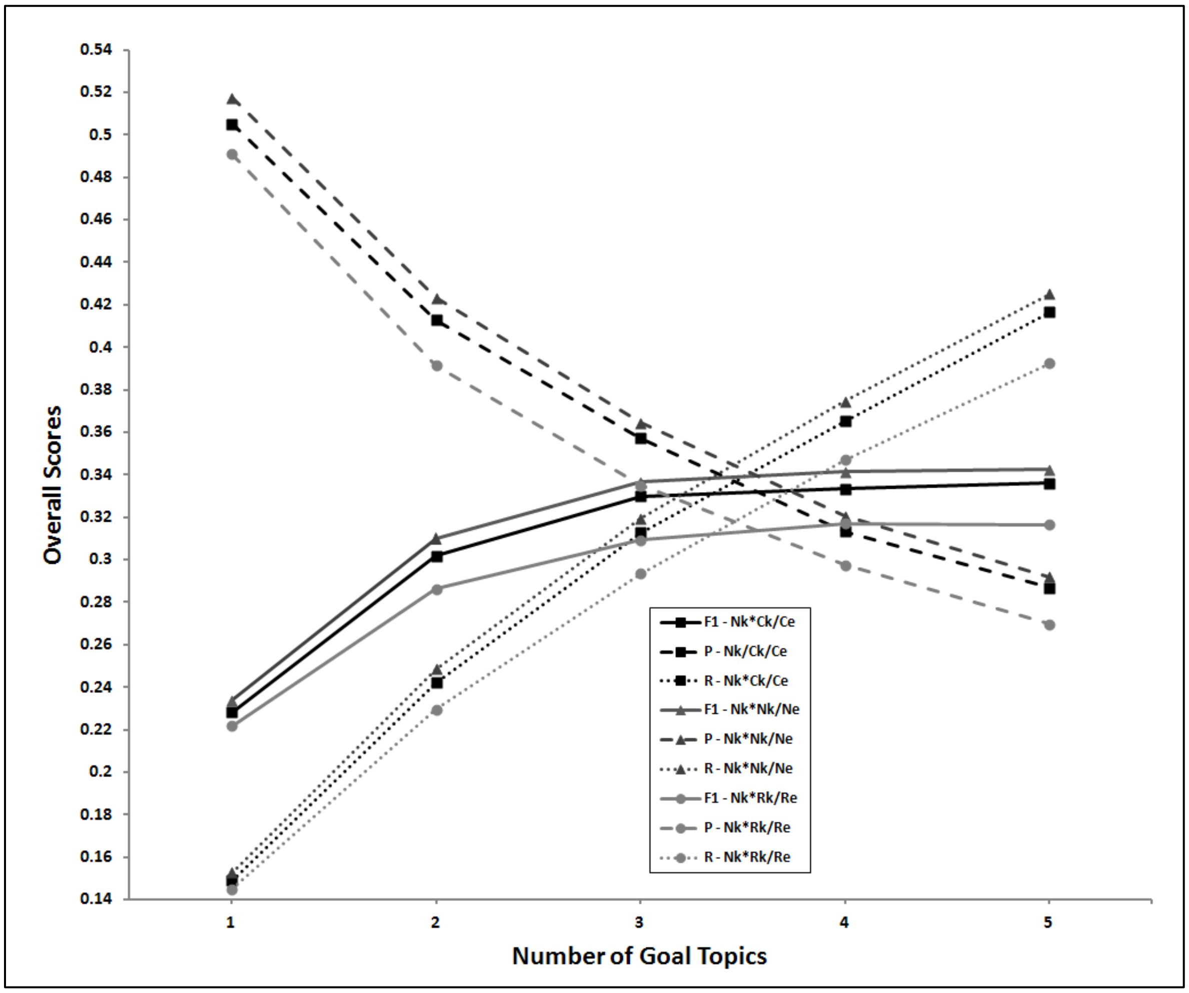


Figure 4.9: Performance metrics for various penalty weight functions[7].

A few observations can be made from Figure 4.9. The first is that the overall result curves of all three variations have the same shape. This means that the system behaves in a very consistent way regardless of the exact formula used. There is no point where one of the results given one equation shoots off in a wildly different range of values from the other two equations. Moreover, while the exact difference in the results between the three equations varies, there is no point where they switch

[7] Overall precision (dashed line), recall (dotted line) and F1 (solid line) using N_k*(C_k/C_e) (dark squares), N_k*(N_k/N_e) (medium triangles), and N_k*(ΣF_k/ΣF_e) (light circles).

and one equation goes from giving worse results than another to giving better results. We can also see that the precision decreases and the recall increases as we increase the number of acceptable goal topics. This result was to be expected: increasing the number of topics returned in the results means that each query is classified in more topics, leading to more correct classification (that increase recall) and more incorrect classifications (that decrease precision). Finally, we can note that the best equation for the penalty weight of extra keywords in entities is consistently the proportion of keywords (N_k / N_e), followed closely by the proportion of characters (C_k / C_e), while the proportion of IDF ($\Sigma F_k / \Sigma F_e$) trails in third position.

It is surprising that the IDF measure gives the worst results of the three, when it worked well in other projects [62]. However, the IDF measure is based on a simple assumption, that a word with low semantic importance is one that is used commonly in most documents of the corpus. In our current system however, the “documents” are article titles, which are by design short, limited to important keywords, and stripped of semantically irrelevant words. These are clearly in contradiction with the assumptions that underlie the IDF measure. We can see this clearly when we compare the statistics of the keywords given in the example in [62] with the same keywords in our system, as we do in Table 4.3. The system in [62] computed its statistics from the entire Wikipedia corpus, including article text, and thus computed reliable statistics which are presented in column 2 and 3; in the example in Table 4.3, the rarely-used company name WWE is found much more significant than the common corporate nouns chief, executive, chairman and headquartered. On the other hand, in our system WWE is used in almost as many titles as executive and has a comparable F_w score, which is dwarfed by the F_w score of chairman and headquartered, two common words that are very rarely used in article titles.

Table 4.3: Comparison of IDF of sample keywords.

Keyword	Num_w^*	F_w^*	Num_w	F_w
WWE	2,705	7.8	657	9.0
Chief	83,977	5.6	1,695	8.1
Executive	82,976	5.8	867	8.7
Chairman	40,241	7.2	233	10.1
Headquartered	38,749	7.1	10	13.2

Finally, we can wonder if the two parts of Equation (3.25) are really necessary (the ordering weight is not used in this experiment at all), especially since the best equation we found for penalty weight repeats the N_k term. To explore that question, we ran the same test again using each part of the equation separately. Figure 4.10 plots the overall F1 using N_k alone in light dotted line with circle markers and N_k / N_e in black dashed line with square markers; it reproduces the overall F1 of N_k * (N_k / N_e) from Figure 4.9 in its medium solid line with triangle markers for comparison. This figure shows clearly that using the complete equation gives better results than using either one of its components.

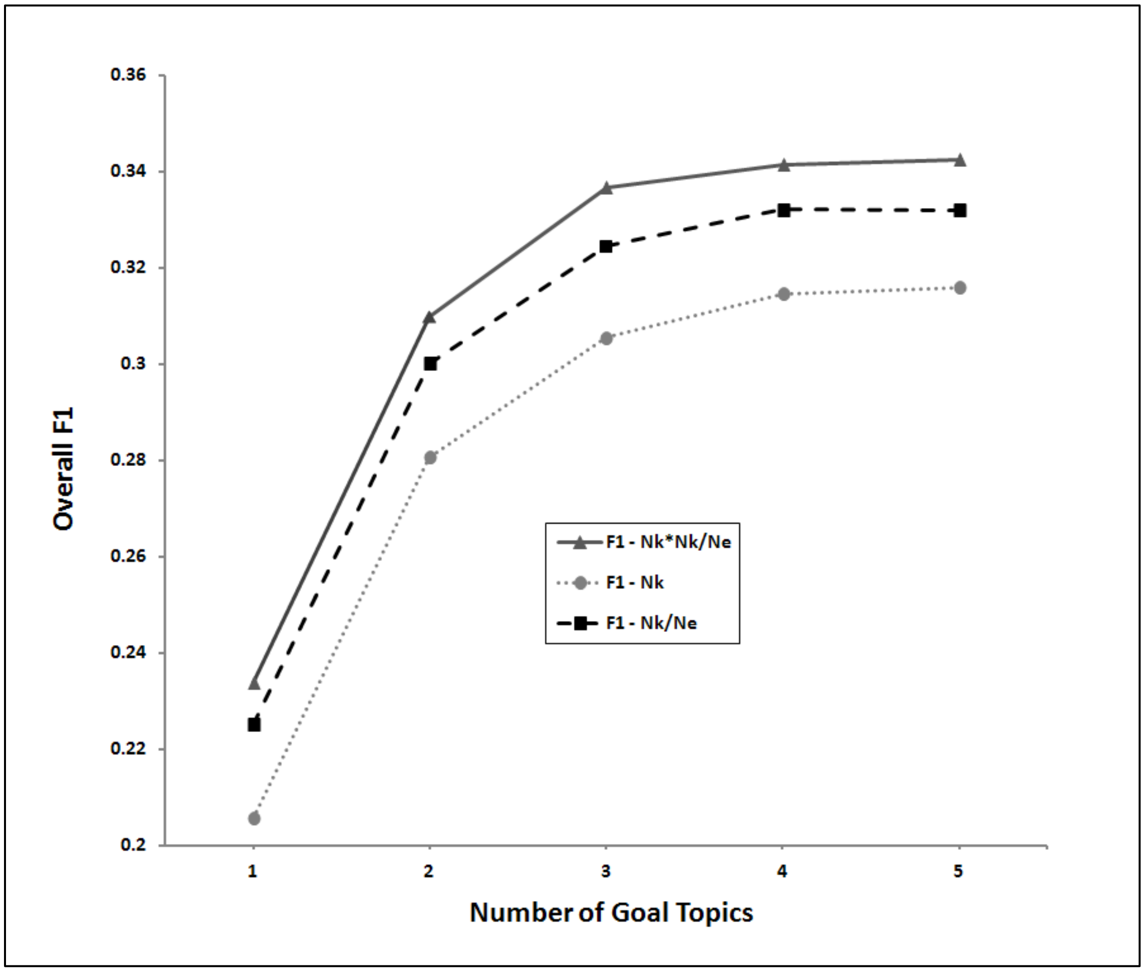


Figure 4.10: Comparison of the components of the entity score.

4.2.3.2 Size of the Base Topic Set

The second aspect of the system we studied comes at the step 7 of the algorithm, when the list of base topics is trimmed down to keep only the most relevant ones. This list will initially contain all topics connected to any entity that contains at least one of the keywords the user specified. As mentioned before, the average number of base topics generated by a query is 3,400 and the maximum is 45,000. These base topics are then used to compute the score of the goal topics, using one of the summations of equations (4.7) to (4.11). This test aims to see if the quality of the results can be improved by limiting the size of the set of base topics used in this summation, and if so what is the approximate ideal size.

For this test, we used $N_k * (N_k / N_e)$ for candidate entity score, the best formula found in the previous test. We again weighted the goal topics using Equation (4.8) and varied the number of returned goal topics from 1 to 5. Figure 4.11 shows the F1 value of the system under these conditions when trimming the list of base topics to 500 (black solid line with diamonds), 100 (light solid line with circles), 50 (medium solid line with triangles), 25 (light dotted line with squares), 10 (black dashed line with squares) and 1 (black dotted line with circles).

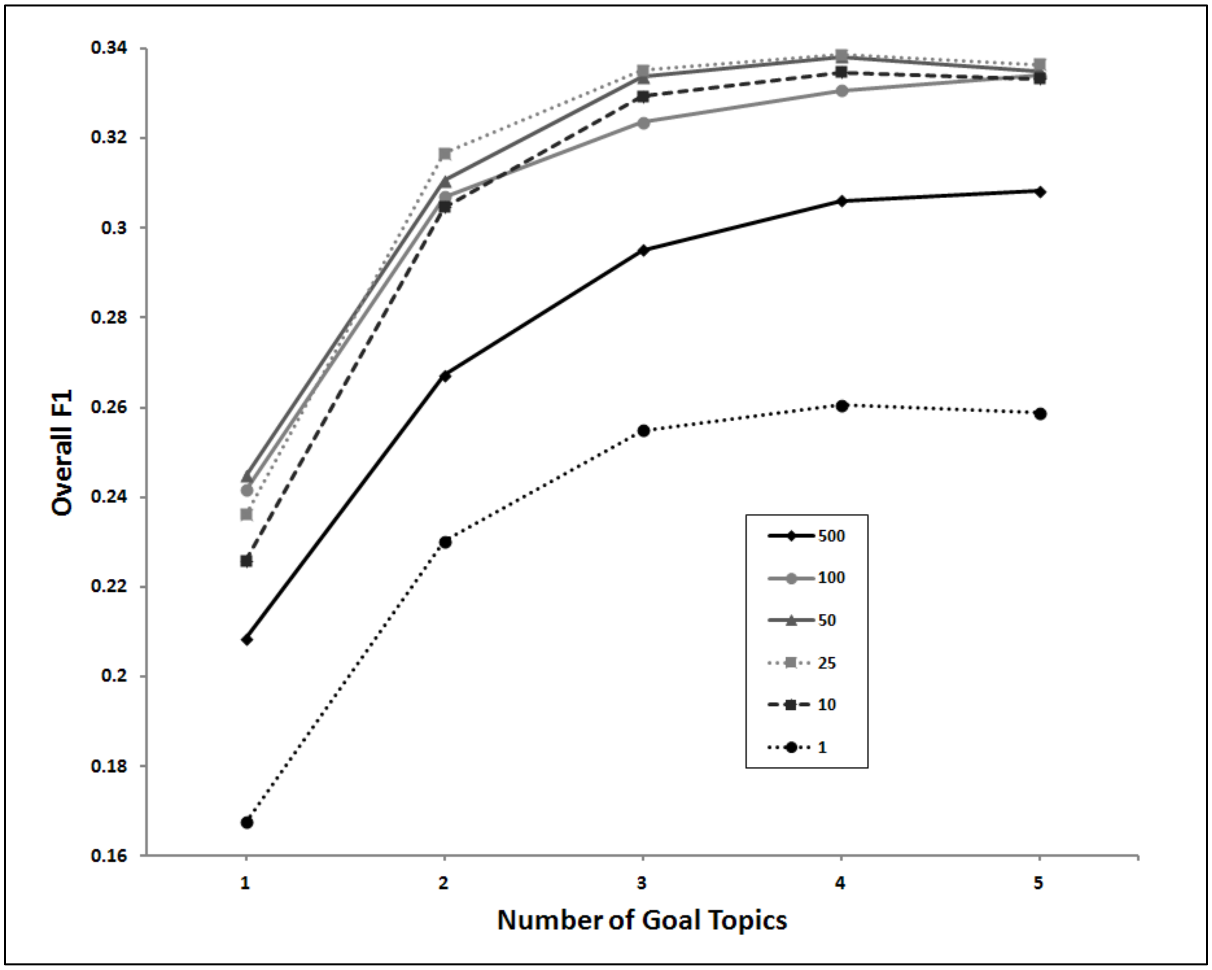


Figure 4.11: Overall F1 using 1 to 500 base topics.

Figure 4.11 shows clearly that the quality of the results drops if the set of base topics is too large (500) or too small (1). The difference in the results between the other four cases is less marked, and in fact the results with 10 and 100 base topics overlap. More notably, the results with 10 base topics start weaker than the case with 100, spike around 3 goal topics to outperform it, then drop again and tie it at 5 goal topics. This instability seems to indicate that 10 base topics are not enough. The tests with 25 and 50 base topics are the two that yield the best results; it thus seems then that the optimal size of the base topic set is in that range. The 25 base topic case outperforms the 50 case, and is the one we will prefer.

It is interesting to consider that in the previous experiment, we used the other alternative we proposed, namely to trim the set based on the density values. The cutoff used was half the density value of the base topic in the set with the highest density; any topic with less than that density value was eliminated. This gave us a set of 28 base topics on average, a result which is consistent with the optimum we discovered in this experiment.

4.2.3.3 Goal Topic Score and Ranking

Another aspect of the system we wanted to study is the choice of equations we can use to account for the base topics' density and distance when ranking the goal topics. The option we used in the previous experiment, to find the nearest goals to any of the retained base topics regardless of their densities, is entirely valid. The alternative we consider here is to rank the goal topics in function of their distance to each base topic and of the density of that base. We proposed five possible equations to mathematically combine density and distance to rank the goal topics. Equation (4.7) considers both distance and density evenly, and the others put progressively more importance on the distance up to Equation (4.11).

To illustrate the different impact of equations (4.7) to (4.11), consider three fictional base topics, one which has a density of 4 and is at a distance of 4 from a goal topic, a second with a density of 4 and a distance of 3 from the same goal topic, and the third with a density of 3 and a distance of 3 from the goal. The contribution of each of these bases to the goal topic in each of the summations is given in Table 4.4. As we can see in this table, the contribution of each base decreases as we move down from Equation (4.7) to Equation (4.11), but it decreases a lot more and a lot faster for the base at a distance of 4. The contribution to the summation of the topic at a distance of 4 is almost equal to that of the topics at a distance of 3 when using Equation (4.7), but is three orders of magnitude smaller when using equation (4.11). That is the result of putting more and more emphasis on distance rather than density: the impact of a farther-away higher-density topic becomes negligible compared to a closer lower-density topic. Meanwhile, comparing the contribution of the two topics of different densities at the same distance shows that, while they are in the same order of magnitude, the higher-density one is always more important than the lower-density one, as we would want.

Table 4.4: Example of the impacts of *h* function.

Equation	*Density 4 Distance 4*	*Density 4 Distance 3*	*Density 3 Distance 3*
(4.7)	1.00	1.33	1.00
(4.8)	0.25	0.44	0.33
(4.9)	0.07	0.20	0.15
(4.10)	0.001	0.01	0.007
(4.11)	4.5×10^{-7}	0.0005	0.0004

We ran tests of our system using each of these five equations. In these tests, we again set the candidate score as $N_k * (N_k / N_e)$, kept the 25 highest-density base topics, and varied from retuning 1 to 5 goal topics. The overall F1 of the variations of the system is presented in Figure 4.12. In this figure, the classification results obtained using Equation (4.7) are shown with a black dashed line with circle markers, Equation (4.8) uses a grey solid line with square markers, Equation (4.9) uses a dashed black line with triangle markers, Equation (4.10) uses a light grey line with circle markers and Equation (4.11) uses a grey line with triangle markers.

We can see from Figure 4.12 that putting too much importance on distance rather than density can have a detrimental impact on the quality of the results: the results using equations (4.10) and (4.11) are the worst of the five equations. Even the results from equation (4.9) are of debatable quality: although it is in the same range as the results of equations (4.7) and (4.8), it shows a clear downward trend as we increase the number of goal topics considered, going from the best result for 2 goals to second-best with 3 goals to a narrow third place with 4 goals and finally to a more distant third place with 5 goals.

Out of curiosity, we ran the same test a second time, but this time keeping the 100 highest-density base topics. These results are presented in Figure 4.13, using the same line conventions as Figure 4.12. It is interesting to see that this time it is Equation (4.9) that yields the best results with a solid lead, not equation (4.8). This indicates a more fundamental relationship in our system: the best summation for the goal topics is not an absolute but depends on the number of base topics retained.

With a smaller set of 25 base topics, the system works best when it considers a larger picture including the impact of more distant topics. But with a larger set of 100 base topics, the abundance of more distant topics seems to generate noise, and the system works best by limiting their impact and by focusing on closer base topics.

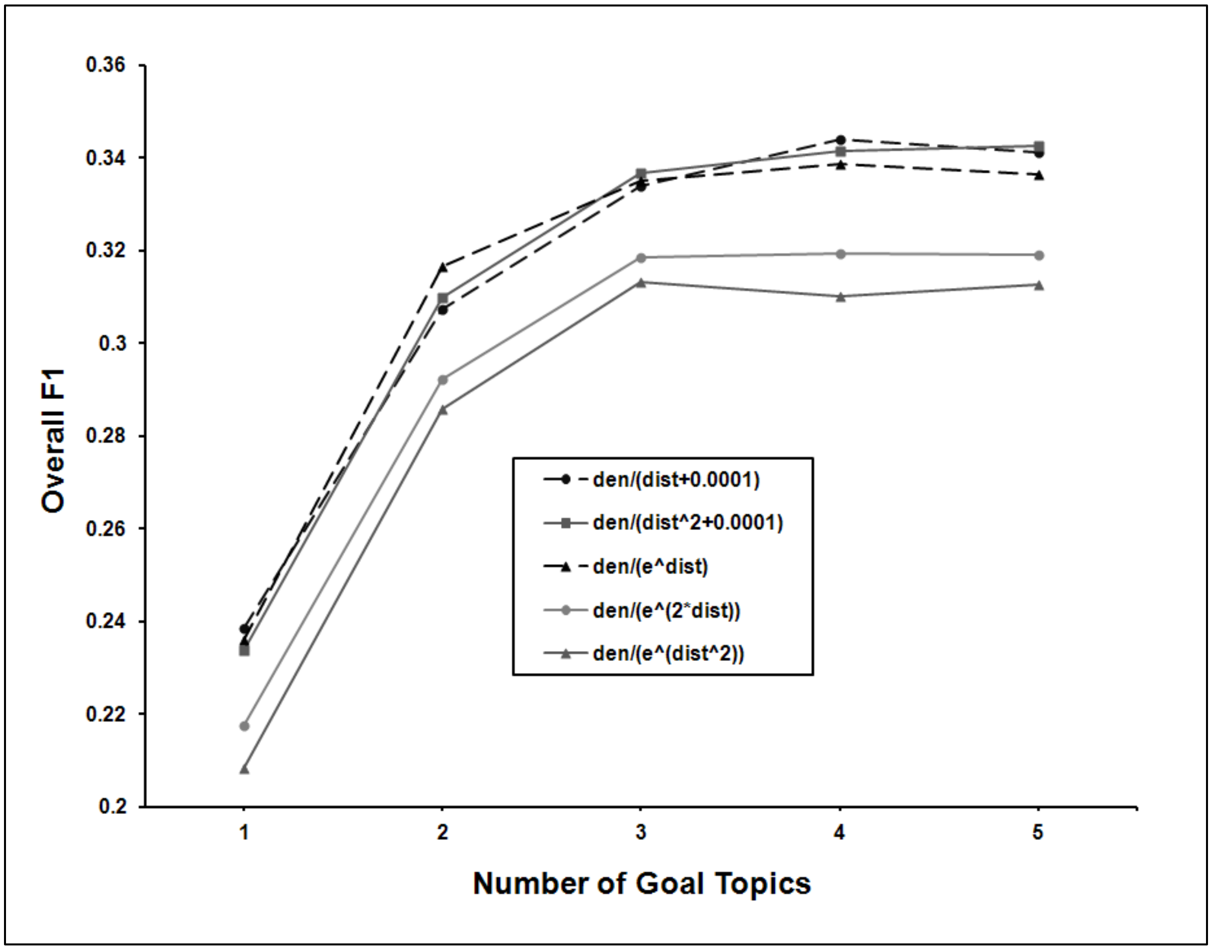


Figure 4.12: Comparison of the impact of the various *h* function equations.

4.2.3.4 Number of Goal Topics Returned

The final parameter in our system is the number of goal topics to return. We have already explained that our preference to return three goal topics is based on a study of human classification – namely, in the KDD CUP 2005 competition, human labelers used on average 3.3 topics per query. Moreover, looking at the F1 figures we presented in the previous subsections, we can see that the curve seems to be exponential, with each extra topic returned giving a lesser increase in F1 value. Returning a fifth goal topic gives the least improvement compared to returning only four goal topics, and in fact in some cases it causes a drop in F1. Returning three topics seems to be at the limit between the initial faster rate of increase of the curve and the later plateau.

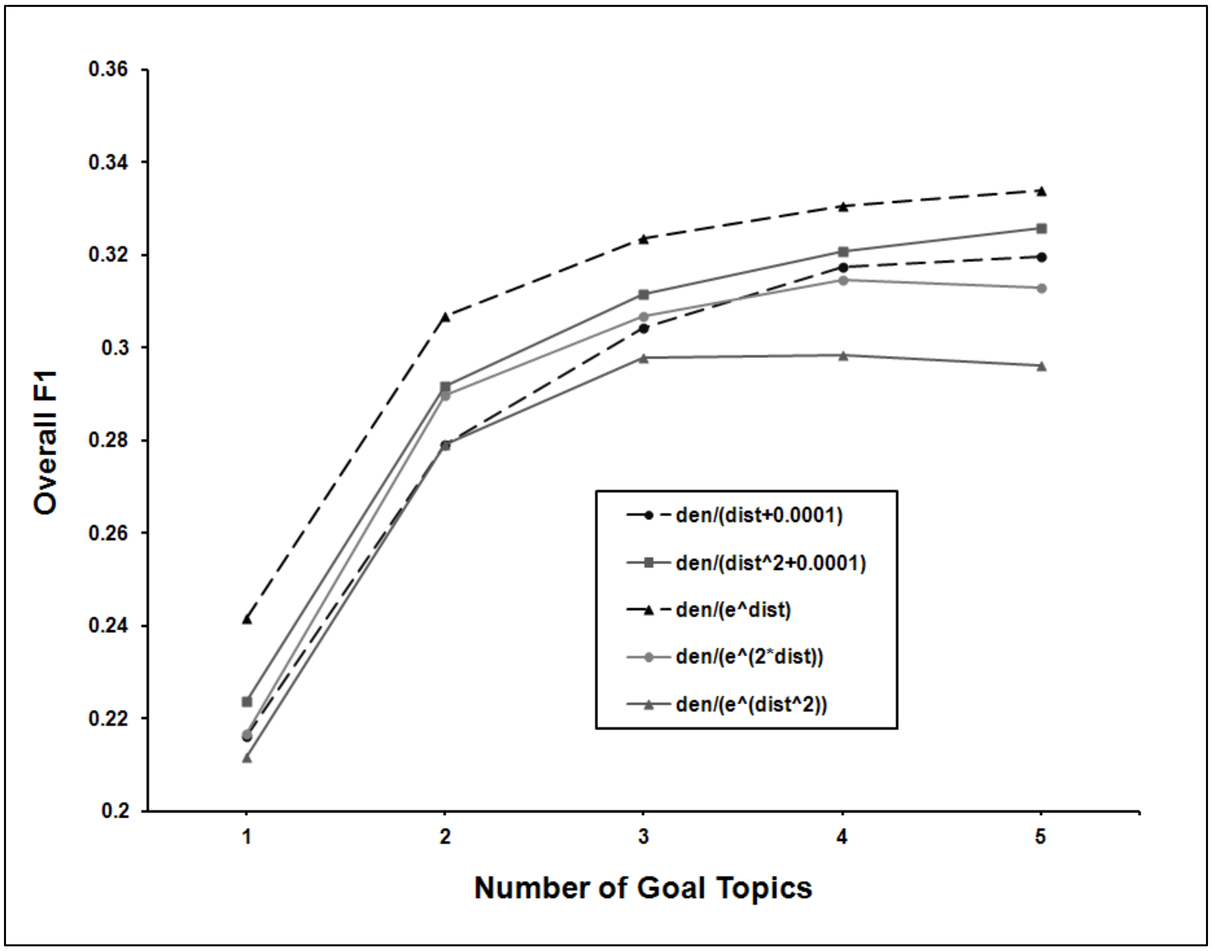


Figure 4.13: Comparison of the impact of the *h* function equations using 100 base topics.

Another way to look at the question is to consider the average score of goal topics at each rank, after summing the densities of the base topics for each and ranking them. If on average the top-ranked topics have a large difference to the rest of the graph, it will show that there exist a robust division between the likely-correct goal topics to return and the other goal topics. The opposite observation, on the other hand, would reveal that the rankings could be unstable and sensitive to noise, and that there is no solid score distinction between the goals our system returns and the others.

For this part of the study, we used the summation of Equation (4.8). We can recall from our discussion earlier that the maximum keyword weight is N_k and the maximum topic density is N_k^2.

Queries in the KDD CUP dataset are at most 10 words long, giving a maximum base topic density of 100. This in turn gives a maximum goal topic score of 1,002,400 using Equation (4.8) and 25 base topics in the case where the distance between the goal topic and each of the base topics is one except for a single base topic at a distance of zero; in other words, the goal is one of the base topics found and all other base topics are immediately connected to it. More realistically, we find in our experiments that the average base topic density is 1.48, and the average distance between a base and goal topic is 5.6 steps, so an average goal topic score using Equation (4.8) would be 1.18.

Figure 4.14 shows the average score of the goal topic at each rank over all KDD CUP queries used in our experiment, obtained using the method described above. This graph shows that the top topic has on average a score of 3,272, several orders of magnitude above the average but still below our theoretical maximum. In fact, even the maximum score we observed in our experiments is only 67,506, very far below the theoretical maximum. This is due to the fact that most base topics are more than a single step removed from the goal topic.

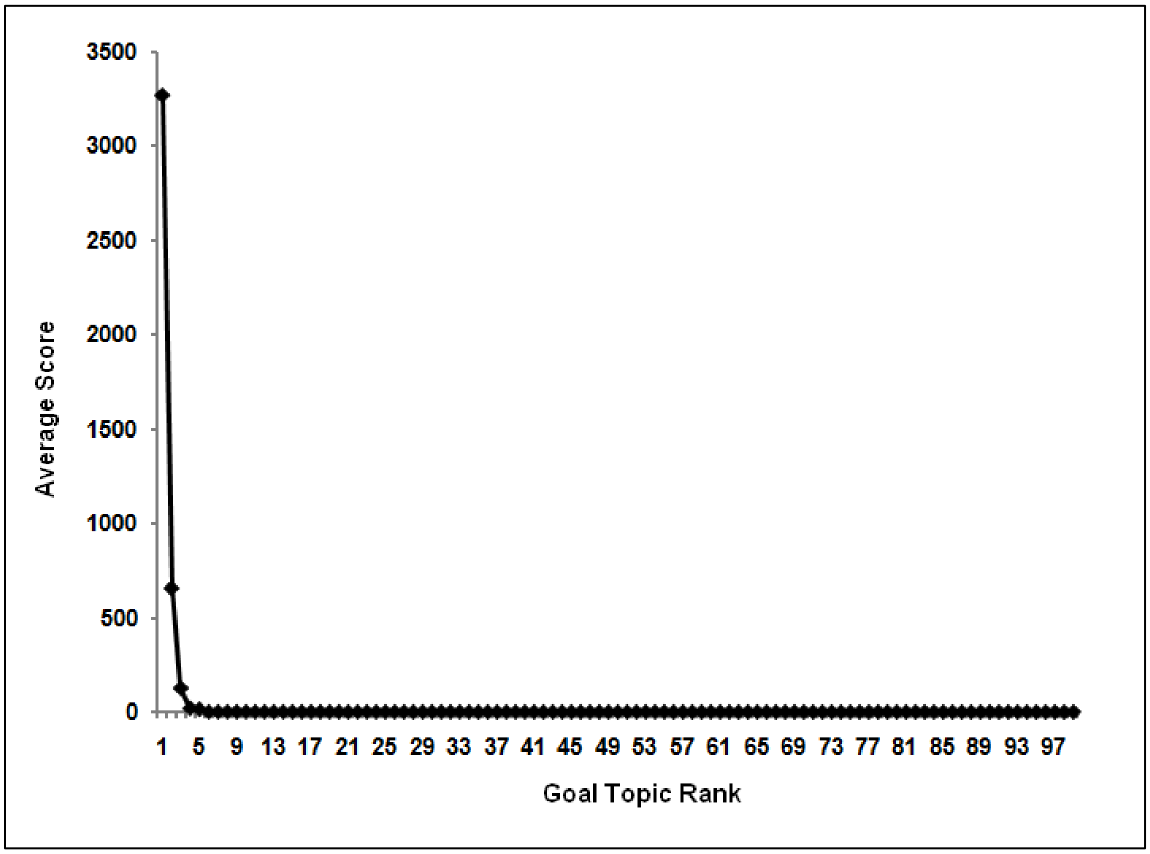


Figure 4.14: Average goal topic score per rank over all KDD Cup queries.

The graph also shows the massive difference between the first three ranks of goal topics and the other 96. The average score goes from 3,272, to 657 at rank 2 and 127 at rank 3, down to 20 and 16 at ranks 4 and 5 respectively, then cover the interval from 2 to 0.7 between ranks 6 and 99. This demonstrates a number of facts. First of all, both the values of the first five goal ranks and the differences between their scores when compared to the other 94 shows that these first ranks are resilient to noise and variations. It also justifies our decision to study the performance of our system using the top 1 to 5 goal topics, and it gives further experimental support to our decision to limit the number of goal topics returned by the classifier to three.

It is interesting to note that the average score of the topics over the entire distribution is 42.53, very far off from our theoretical average of 1.18. However, if we ignore the first three ranks, whose values are very high outliners in this distribution, the average score becomes 1.62. Moreover, the average score over ranks 6 to 99 is 1.28. Both of these values are in line with the average we expected to find.

4.2.3.5 The Optimal System

After having performed the above test, we are ready to put forward the optimal classifier, or the one that combines the best features from the options we have studied. This classifier uses $N_k * (N_k / N_t)$ for candidate entity score, selects the top 25 base topics, ranks the goal topics using the commonality summation formula of Equation (4.8), and returns the top-three topics ranked. The results we obtain with that system are presented in Table 4.5, with other KDD CUP competition finalists reported in [53] for comparison. As can be seen from Table 4.5, our system performs well above the competition average, and in fact ranks in the top-10 (7th place) of the competition in F1 and in the top-5 (2nd place) in precision. This is also a considerable improvement over the results obtained in the previous experiment using the iterative exploration method. For comparison, system #22, which won the first place in the competition, achieved the best results by querying multiple web search engines and aggregating their results [54]. Our method may not perform as well right now, but it offers the potential for algorithmic and knowledge-base improvements that goes well beyond those of a simple aggregate function, and is not dependent on third-party commercial technology.

We also found in our results that 47 of the 800 test queries were not classified at all, because the algorithm failed to select any base topics at all. This situation occurs when no entities featuring query words can be found. These queries are all single words, and that word is either an uncommon abbreviation (the query “AATFCU” for example), misspelled in an unusual way (“egyptains”), an erroneous compounding of two words (“contactlens”), a rare website URL, or even a combination of the above (such as the misspelled URL “studioeonline.com” instead of “studioweonline.com”). These are all situations that occur with real user search queries, and are therefore present in the KDDCUP dataset. It is worth noting that Wikipedia titles which are the source of the entities in the semantic graph include common cases of all these errors, so that only the 5.9% most unusual cases lead to failure in our system.

Table 4.5: Semantic distance topic identifier performance in baseline standings[8].

System	F1 Rank	Overall F1	Precision Rank	Overall Precision
KDDCUP #22	1	0.4444	N/A	0.4141
KDDCUP #37	N/A	0.4261	1	0.4237
KDDCUP #21	6	0.3401	2*	0.3409
Our system	**7**	**0.3366**	**2**	**0.3643**
KDDCUP #14	7*	0.3129	N/A	0.3173
KDDCUP Mean		0.2353		0.2545
KDDCUP Median		0.2327		0.2446

4.2.3.6 Case Study

It could be interesting to study a specific example, to see the system’s behavior step by step. We chose for this purpose to study a query for “internet explorer” in the KDDCUP set. This query was manually classified by the competition’s three labelers, into the KDDCUP categories “Computers\Software; Computers\Internet & Intranet; Computers\Security; Computers\Multimedia; Information\Companies & Industries” by the first labeler, into “Computers\Internet & Intranet; Computers\Software” by the second labeler, and into “Computers\Software; Computers\Internet & Intranet; Information\Companies & Industries” by the third labeler.

The algorithm begins by identifying a set of relevant base topics using the calculated candidate entities and then weighting them using the density function. For this query, our algorithm identifies 1,810 base topics, and keeps the 25 highest-density ones, breaking the tie for number 25 by considering the number of entities pointing to the topics. For any two-word query, the maximum candidate entity score value is 2, and the maximum base topic density value is 4. And in fact, we find that 8 topics receive this maximum density, including some examples we listed in Table 4.6. We can see from these examples that the top-ranked base topics are indeed very relevant to the query.

[8] * indicates competition systems that would have been outranked by ours.

Examining the entire set of base topics reveals that the density values drop to half the maximum by rank 33, and to a quarter of it by rank 37. The density value continues to drop as we go down the list: the average density of a base topic in this example is 0.40 which corresponds to rank 660, by the middle of the list at rank 905 the density is 0.33, and the final topic in the list has a density of only 0.05. It can also be seen from the samples in Table 4.6 that the relevance of the topics to the query does seem to decrease along with the density value. Looking at the complete list of 1,810 base topics, we find that the first non-software-related topic is “Exploration” at rank 41 with a density of 1. But software-related topics continue to dominate the list, mixed with a growing number of non-software topics, until rank 354 (density of 0.5 and 1 entity pointing to the topic) where non-computer topics begin to dominate. Incidentally, the last software-related topic in the list is “United States internet case law”, at rank 1791 with a density of 0.11.

Table 4.6: Base topic samples for "internet explorer" query case study.

Topic	**Rank**	**Density**	**Entities#**
Internet Explorer	1	4	36
Internet history	2	4	32
Windows web browsers	3	4	20
Microsoft criticisms and controversies	8	4	4
HTTP	25	2.67	5
Mobile phone web browsers	26	2.67	4
Cascading Style Sheets	33	2	5
Internet	37	1	17
PlayStation Games	660	0.40	2
Islands of Finland	905	0.33	1
History of animation	1811	0.05	1

The next step of our algorithm is to rank the 99 goal topics using the sum of density values in Equation (4.8). Sample rankings are given in Table 4.7. This table uses the Wikipedia goal topic labels; the matching KDDCUP categories can be found in Appendix A. We can see from these results that the scores drop by half from the first result to the fourth one. This is much less drastic than the drop we observed on average in Figure 4.14, but is nonetheless consistent as it shows a quick drop from a peak over the first three ranks and a long and more stable tail over ranks 4 to 99.

It is also encouraging to see that the best two goal topics selected by our system correspond to “Computers\Internet & Intranet” and “Computers\Software”, the only two categories to be picked by all three KDDCUP labelers. The fourth goal corresponds to “Online Community/Other” and is the first goal that is not in the KDDCUP “Computer/” category, although it is still strongly relevant to the query. Further down, the first goal that corresponds neither to a “Computers/” nor “Online Community/” category is Technology (“Information\Science & Technology”) at rank 16, which is still somewhat related to the query, and the first truly irrelevant result is Magazines (“Living\Book & Magazine”) at rank 18 with a little over a quarter of the top topic’s score. Of the categories picked by labelers, the one that ranked worst in our system was “Information\Companies & Industries” at rank 30. All the other categories they identified are found in the top-10 results of our system.

Table 4.7: Goal topic samples for "internet explorer" query case study.

Goal Topic	Rank	Score
Internet	1	11.51
Software	2	10.09
Computing	3	8.03
Internet culture	4	5.63
Websites	5	4.97
Technology	16	3.54
Magazines	18	3.39
Industries	30	2.86
Law	49	2.54
Renting	99	1.30

4.2.3.7 New Data and Final Tests

In order to show that our results in Table 4.5 are general and not due to picking the best system for a specific dataset, we ran two more tests of our system with two new datasets. The first dataset is a set of 111 KDD CUP 2005 queries classified by a competition judge. This set was not part of the 800 test queries we used previously; it was a set of queries made available by the competition organizers to participants prior to the competition, to develop and test their systems. Naturally, the queries in this set will be similar to the other KDD CUP queries, and so we expect similar results.

The second dataset is a set of queries taken from the TREC 2007 Question-Answering (QA) track [106]. That dataset is composed of 445 questions on 70 different topics; we randomly selected three questions per topic to use for our test. It is also worth noting that the questions in TREC 2007 were designed to be asked sequentially, meaning that a system could rely on information from the previous questions, while our system is designed to classify each query by itself with no query history. Consequently, questions that were too vague to be understood without previous information were disambiguated by adding the topic label. For example, the question ‘Who is the CEO?’ in the series of questions on the company 3M was rephrased as ‘Who is the CEO of 3M?’ Finally, two volunteers independently labeled the questions to KDD CUP categories in order to have a standard to compare our system’s results to using equations (4.1) to (4.6). The TREC dataset was selected in order to subject our system to very different testing conditions: instead of the short keyword-only KDD CUP web queries, TREC has long and grammatically-correct English questions.

The results from both tests are presented in Table 4.8, along with our system’s development results already presented in Table 4.5 for comparison. These results show that our classifier works better with the test data than with the training data it was developed and optimized on. This counter-intuitive result requires explanation.

Table 4.8: Topic identification performance results for new test sets.

Query Set	**Overall F1**	**Overall Precision**	**Overall Recall**
KDDCUP 111	0.3636	0.4254	0.3175
TREC	0.4639	0.4223	0.5267
KDDCUP 800	0.3366	0.3643	0.3195

The greatest difference in our results is on recall, which increases by over 20% from the training KDDCUP test to the TREC test. Recall, as presented in Equation (4.2), is the ratio of correct topic labels identified by our system for a query to the total number of topic labels the query really has. Since our classifier returns a fixed number of three topics per query, it stands to reason that it cannot achieve perfect recall for a query set that assigns more than three topics, and that it can get better recall on a query set that assigns fewer topics per query. To examine this hypothesis, we compared the results of five of the labelers individually: the three labelers of the KDDCUP competition and the two labelers of the TREC competition (the 111 KDDCUP demo queries, having been labeled by only one person, were not useful for this test). Specifically, we looked at the average number of topics per query each labeler used and the recall value our system achieved using that query set. The results, presented in Table 4.9, show that our intuition is correct: query sets with less topics per query lead to higher recall, with the most drastic example being the increase of 1.5 topics per query between KDDCUP labelers 2 and 3 that yielded a 10% decrease in recall. However, it also appears from that table that the relationship does not hold across different query sets: KDDCUP labeler 2 assigns less labels per query that TREC labeler 2 but still has a much lower recall.

Table 4.9: Comparison of topic number and recall for different labelers.

Query Set	**Average Number of Topics**	**Recall**
TREC Labeler 1	1.93 ± 0.81	0.5443
TREC Labeler 2	2.91 ± 0.92	0.5090
KDDCUP Labeler 2	2.39 ± 0.93	0.3763
KDDCUP Labeler 1	3.67 ± 1.13	0.3076
KDDCUP Labeler 3	3.85 ± 1.09	0.2747

Next, we can contrast the two KDDCUP tests: they both had nearly identical recall but the new data gave a 6% increase in precision. This is interesting because the queries are from the same datasets, they are web keyword searches of the same average length, and the correct categorization statistics are nearly identical to those of Labeler 3 so we would actually expect the recall to be lower than it ended up being. An increase in both precision and recall can have the same origin in equations (4.1) and (4.2): a greater proportion of correct topics identified by our classifier. But everything else being equal, this would only happen if the queries themselves were easier for our system to understand. To verify this hypothesis, we checked both query sets for words that are unknown in our system. As we explained previously, a lot of these words may be rare but simple typos ("egyptains") or missing spaces between two words ("contactlens"), and while they are unknown and ignored in our system their meaning is immediately obvious to the human labelers. The labelers thus have more information to classify the queries, which makes it inherently more difficult for our system to generate the same classification. Upon evaluation of our data, we find that the KDDCUP set of 800 queries features about twice the frequency of unknown words of the set of 111 queries. Indeed, 10.4% of queries in the 800-query set have unknown words and 4.4% of words overall are unknown, while only 5.4% of queries in the 111-query set have unknown words and only 2.5% of words in that set are unknown. This is an important difference between the two query sets, and we believe it explains why the 111 queries are more often classified correctly. It incidentally also indicates that an automated corrector should be incorporated in the system in the future.

The better performance of our system on the TREC query set can be explained in the same way. Thanks to the fact that set is composed of correct English questions, it features even fewer unknown words: a mere 0.4% of words in 1.9% of queries. Moreover, for the same reason, the queries are much longer: on average 5.3 words in length after stop word removal, compared to 2.4 words for the KDDCUP queries. This means that even if there is an unknown word in a query, there are still a lot of other words in the TREC queries for our system to make a reasonably good classification.

Differences in the queries aside, it does not appear to be major distinctions, much less setbacks, when using our classifier on new and unseen datasets. It seems robust enough to handle new queries in a different spread of domains, and to handle both web-style keyword searches and English questions without loss of precision or recall.

Finally, it could be interesting to determine how our classifier's performance compares to that of a human doing the same labeling task. Query classification is a subjective task: since queries are short and often ambiguous, their exact meaning and classification is often dependent on human interpretation [107]. It is clear from Table 4.9 that this is the case for our query sets that human labelers do not agree with each other on the classification of these queries. We can evaluate the human labelers by computing the F1 of each one's classification compared to the others in the same dataset. In the case of the KDDCUP data, the average F1 of human labelers is known to be between 0.4771 and 0.5377 [53], while for our labeled TREC data we can compute the F1 between the two human labelers to be 0.5605. This means our system has between 63% and 71% of a human's performance when labeling the KDDCUP queries, and 83% of a human's performance when labeling the TREC queries. It thus appears that by this benchmark, our classifier again performs better on the TREC dataset than on the KDDCUP one. This gives further weight to our conclusion that our system is robust enough to handle very diverse queries.

4.3 Speech Recognition

The final part of this chapter is a brief study of the speech recognition application firstly introduced in Section 3.8. Here we start by describing the problem specification in more details. Then, a variation of the algorithm presented in Section 3.8 is explained along with an example for a better clarification of algorithm mechanism. Finally, the experimental results of the algorithm using real test data are presented and compared with baseline methods.

Speech decoders are becoming prominent in an increasing number of real-world applications and are required to become more robust and more reliable. Nowadays, speech transcription is being used ubiquitously. The requirement for highly accurate decoders has increased substantially. Most researchers agree that one of the most promising approaches to the problem of reducing word error rate (WER) in speech transcription is to combine two or more speech decoders and compile a new composite more accurate output [108]. The Recognizer Output Voting Error Reduction (ROVER) is the most widely used technique in this regard. However, its performance has stagnated over the past few years mainly due to the limitations of the voting algorithms used. These shortcomings include the use of unreliable word level confidence scores of decoders as well as randomly-broken ties when the frequency of occurrences is used during the voting. This experiment proposes a novel voting scheme for the ROVER combination procedure which relies on the entity graph as patterns to select the winner token from the different hypotheses of the decoders.

4.3.1 ROVER Procedure

ROVER [109] is a system developed at NIST in 1997 to produce a composite of decoder outputs when the outputs of multiple automatic speech recognizers (ASR) are available. The goal is to produce a lower WER in the composite output. This is done through a voting mechanism to select the winner word from among the different decoder outputs. ROVER is a two-step process as shown in Figure 4.15. It starts by combining the multiple decoder outputs into a single minimal-cost word transition network (WTN) through dynamic programming. The confidence scores of the recognizers are shown in parentheses in the WTN of Figure 4.15. In the second step, the resulting WTN is browsed by a voting process which selects the best output sequence out of potential word sequences presented by different ASRs. Three voting mechanisms have been presented in [109]. At each location in the composite WTN, a score is computed for each word using Equation (4.12) where *i* is the current location in the WTN, N_s is the total number of combined ASR systems, $N(w, i)$ is the frequency of word *w* at the position *i*, and $C(w, i)$ is the confidence value for word *w* at the position *i*.

$$S_w = \alpha\left(\frac{N(w,i)}{N_s}\right) + (1-\alpha)\cdot C(w,i) \tag{4.12}$$

The parameter α is set to be the trade-off between using word frequency and confidences. In the case there is an insertion or deletion, the NULL transition, noted as @, in the word transition network will have the confidence conf(@). A training stage is therefore needed to optimize both the parameter and the NULL transition confidence value. This is commonly done through grid-based searching. The three voting schemes are frequency of occurrence, frequency of occurrence and average word confidence, and frequency of occurrence and maximum confidence. The first scheme only uses occurrences to select the winning word at each WTN slot. However, ties tend to occur frequently and they are randomly broken. The remaining two schemes rely on word confidences to overcome the problem of arbitrarily broken ties. The fact that ROVER's voting relies on confidences makes it a vulnerable technique. It is not safe to assume that the word confidences are reliable. Much research [110] is still pursued in attempts to come up with a robust and effective technique to provide a decent confidence measure for Large Vocabulary Continuous Speech Recognition (LVCSR). Even in [109], neither algorithm that relies on a confidence score achieved a significant error reduction compared to the frequency-based voting algorithm. Furthermore, ROVER is unable to outvote the erroneous ASR systems when only one single ASR is providing the correct output. The scoring schemes make it difficult to boost a single ASR output since both occurrence and confidence are used to score each word at a specific location.

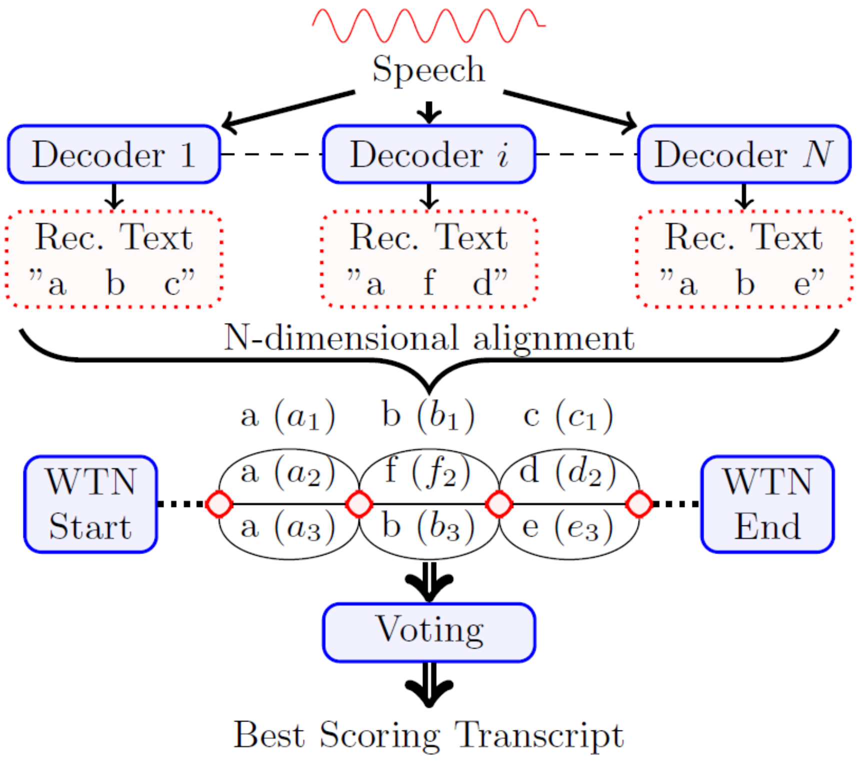


Figure 4.15: ROVER procedure.

4.3.2 Entity-based Voting Algorithm

The proposed approach in this experiment relies on entity-to-WTN pattern matching in order to intelligently select the winner token at each location of the WTN. The assumption that underlies our work is that there exist multi-word statements that a speaker is likely to use and that we can improve the voting stage within the ROVER procedure by recognizing these multi-word statements. In this context, multi-word statements refer to a variety of entities such as long nouns (e.g. the honorable Member of Parliament) or proper names (e.g. Harvard University). Figure 4.16 presents the steps of our proposed approach. The goal of the algorithm is to browse the WTN and select the best word at each slot (called token) from different parallel sentences provided by each ASR system. The words which are selected as winners are tagged as *WINNER*. The algorithm continues until it can find a winner for all the slots and, therefore, generate one final sentence as the output result.

```
Input: WTN
1.   Tag all common tokens as WINNER
2.   Find all the candidate entities matching any n-gram word
     combination in WTN with at least two consecutive words
3.   While slots without a WINNER token exist do:
4.       While any candidate entity remains do:
5.           Update the candidate entity list by discarding all
             entities that do not match any WTN slot without a
             WINNER tag
6.           Score and rank all candidate entities
7.           Select the highest score candidate entity as winner
8.           Tag tokens matching the winner entity n-grams words as
             WINNER
9.           Remove the winner entity from the candidate entity list
10.      End while
11.      Use frequency of occurrences to select WINNER in the
         remaining slots, if any
12.  End while
```

Figure 4.16: Entity-based voting algorithm.

In step 1 of the algorithm, we browse the WTN to locate all slots in which the same token has been recognized by all speech decoders. Therefore, we can tag that token as the *WINNER*. In other words, if all speech decoders agreed on the same token at a given slot, we suppose it has been correctly recognized and hence, we tag it as *WINNER*. Step 2 finds all the candidate entities which match any *n*-gram of consecutive tokens within the WTN. There is a requirement that the candidate entity should match at least two consecutive tokens in the WTN. Moreover, at least one of the tokens featured in the candidate entity should not already be marked as *WINNER*. These conditions ensure that the found candidate entities can be used to discover new *WINNER* tokens. Once all eligible candidate entities are found, a loop will be created until there is no slot without a *WINNER* token. The next nested loop iterates over found candidate entities list and process them until no candidate entities remain in the list. At the start of the loop in step 5, the list of candidate entities gets updated and any candidate entity which does not contribute to finding new *WINNER* tokens is discarded. In other words, only candidate entities which feature at least one word matching a WTN slot without a *WINNER* tag will be kept.

Step 6 is a crucial part of the algorithm. In this step, all candidate entities are scored and then sorted descending over their scores. The equation used here to score the candidate entities is a little different than score equations presented so far. Equation (4.13) shows the scoring formula used for scoring candidate entities.

$$S(e_c) = \left(\sum W_k\right) \times \frac{N_k - N_x}{N_e} \tag{4.13}$$

The above equation is a transformation of basic candidate entity score introduced by Equation (3.25). The first term (summation) represents an extended version of the word featuring weight while the second term (fraction) represents the new penalty weight. In this equation, W_k is the number of occurrences of word k in its slot over different ASR systems. N_k is the number of consecutive words found in the entity i.e. it is equal to n as in n-gram found in the entity. N_x is the number of false matches i.e. the number of words within the n-gram which match tokens at slots where a *WINNER* tag is attributed to a different token. Finally, N_e is, as before, equal to the total number of entity words. An important note here is the impact of the NULL transition in Equation (4.13). The NULL token has a weight, which is its occurrences at the slot but it does not count as a word. It can thus contribute to the value of W_k, but not to N_k or N_e.

Once all candidate entities are scored, the top scored candidate entity is selected as the winner entity and the token words featured in the winner entity are selected as *WINNER* tokens. Afterwards, the winner entity is removed from the candidate entity list so that other entities are selected for other remaining slots. The loop continues until all candidate entities are exhausted and no more remaining. In the case that there are still slots available without a *WINNER* tag, step 11 indicates that the token with the most frequency of occurrences is selected at the *WINNER* token. In other words, the token with the highest W_k is the winner word. If ties occur, they are broken randomly.

The next subsection presents an example which illustrates all the above steps. It is notable that the algorithm explained above might be slower compared to the classic ROVER voting algorithm because of the additional processing it requires. For offline applications, however, this should not be an issue. For real-time applications, the algorithm can be optimized in a number of ways: for instance, by pruning the infrequent entities from the semantic graph, by caching frequently-used entities, and by making use of the text indexing tools available in many commercial database systems.

4.3.3 Illustrative Example

In order to better illustrate the steps in Figure 4.16, we present a working example. Let us assume that a given speech waveform is transcribed by three different speech decoders. The resulting WTN is given in Figure 4.17. Per step 1 of the algorithm, the token "The" at the first slot is a common token and, therefore, a *WINNER*.

Now let us assume for the sake of this example that only four candidate entities are found for this WTN. These entities, along with their initial scores, are given in Table 4.10. Notice that the first entity matches three words: "united", "states", and "ink". But only the first two words are consecutive. Consequently, only "united states" is matched with the WTN and "ink" is ignored. Notice also that the "boarding school" entity gets an extra word weight because of the NULL transition between "boarding" and "school". Finally, note that none of the candidate entities cover the initial slot 0, the only position for which a *WINNER* token is identified, and consequently $N_x = 0$ for all candidate entities.

At the first iteration, the highest-scored candidate entity is "boarding school". Therefore, the tokens "boarding", "NULL", and "school" at slots 3, 4, and 5 are tagged as *WINNER*. This entity is removed from the list, before the second iteration begins. The next iteration starts by updating the scores of the remaining three candidate entities. The updated list of candidate entities is shown in Table 4.11. Only one entity has its score changed: "states board of education", which has a false-matched token (N_x) at slot 3 as "board" where "boarding" was tagged as *WINNER* during the previous iteration. Therefore, the highest-scored candidate entity is "united states of ink" in this iteration. The words "united" and "states" are thus tagged as *WINNER* at slot 1 and 2 respectively. The remaining candidate entities only match slots with winner tokens. Therefore, no more candidate entities are usable and they all get discarded per step 5 of the algorithm. This means that the algorithm deals with slot 6 using frequency of occurrence per step 11, thus marking "band" with frequency 2 a *WINNER* token over "bland" with a frequency of 1. The composite final sentence is thus "the united states boarding school band".

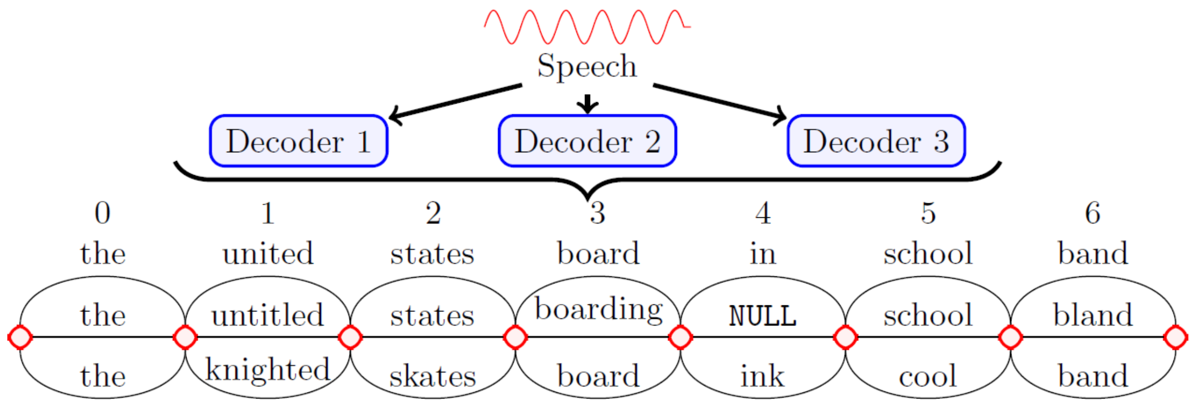


Figure 4.17: A sample WTN for three decoders.

Table 4.10: Initial candidate entities and their scores.

Candidate Entities	**Matched Slots**	**Score ($S(e_c)$)**
"united states of ink"	1 – 2	(1 + 2) × (2 - 0)/4 = 1.5
"knighted skates inc."	1 – 2	(1 + 1) × (2 - 0)/3 = 1.33
"states board of education"	2 – 3	(2 + 2) × (2 - 0)/4 = 2
"boarding school"	3 – 5	(1 + 1+ 2) × (2 - 0)/2 = 4

Table 4.11: Remaining candidate entities and updated scores in the second iteration.

Candidate Entities	**Matched Slots**	**Score ($S(e_c)$)**
"united states of ink"	1 – 2	(1 + 2) × (2 - 0)/4 = 1.5
"knighted skates inc."	1 – 2	(1 + 1) × (2 - 0)/3 = 1.33
"states board of education"	2 – 3	(2 + 2) × (2 - 1)/4 = 1

4.3.4 Datasets and Result

The tests were conducted using the transcription output of the Carnegie Mellon University Sphinx 4 decoder. In terms of data, we have considered the English Broadcast News Speech (HUB4) testing framework [111]. This corpus is composed of both speech data (LDC98S71) and transcripts (LDC98T28). It is a total of 97 hours of 16000 Hz recordings from radio and television news broadcasts. Transcriptions of this HUB4 corpus have been used to train the language model (named as LM-98T28 hereafter). Two other freely available language models were used: an open-source model for broadcast news transcriptions from CMU [112], referred to as LM-BN99 hereafter, and another language model created from the English Gigaword corpus [113], referred to as LM-GIGA hereafter. In order to simulate outputs of speech decoders from several sites, we have loaded the Sphinx 4 decoder with different language models. A total of three configurations have been set up: Sphinx 4 loaded with LM-98T28 (s4-LM-98T28), Sphinx 4 loaded with LM-BN99 (s4-LM-BN99), and Sphinx 4 loaded with LM-GIGA (s4-LM-GIGA). All possible binary combinations were carried out. Table 4.12 reports the setup. Notice here that the order of combination matters. This is an inherent problem of ROVER's WTN building stage. Because of the use of different language models, a normalization step is required to standardize the output of the different speech decoders. In other words, the same word can be written in different ways and, therefore, all these variations have to be unified under a single form. Examples of these issues include acronyms like CNN which can be written as c. n. n. (three distinct letters), c.n.n. (one single word), cnn (one single word), c n n (three letters), and capitalization, such as Vote, voTe, vote, etc. Moreover, the positions of NULL tokens are determined according to the order of transitions which means the each combination order causes different NULL tokens to be inserted into transitions.

It is worth mentioning that the Sphinx 4 confidence scores were not reliable. Consequently, using these scores in the voting algorithm causes no change in the outcome. For this reason, we only used the frequency-based voting in the reminder our tests. In terms of candidate entities, we have used the list of all titles in Wikipedia that are two or more words in length from the October 2010 version of the encyclopedia. This condition created a set of 6,479,950 entities ranging from bi-grams up to 45-grams, averaging 3-grams in length.

The baseline performance is obtained through the use of the frequency of occurrence voting scheme. The ROVER baseline WER is given in Table 4.13. The First row in this table represents the different binary combinations previously defined in Table 4.12.

Table 4.12: Decoders combinations.

ID	Combination Configuration
C1	s4-LM-GIGA - s4-LM-98T28
C2	s4-LM-GIGA - s4-LM-BN99
C3	s4-LM-98T28 - s4-LM-GIGA
C4	s4-LM-98T28 - s4-LM-BN99
C5	s4-LM-BN99 - s4-LM-98T28
C6	s4-LM-BN99 - s4-LM-GIGA

Table 4.13: ROVER baseline WERs

Test	C1	C2	C3	C4	C5	C6
WER	37.12	36.45	37.32	35.51	32.45	33.10

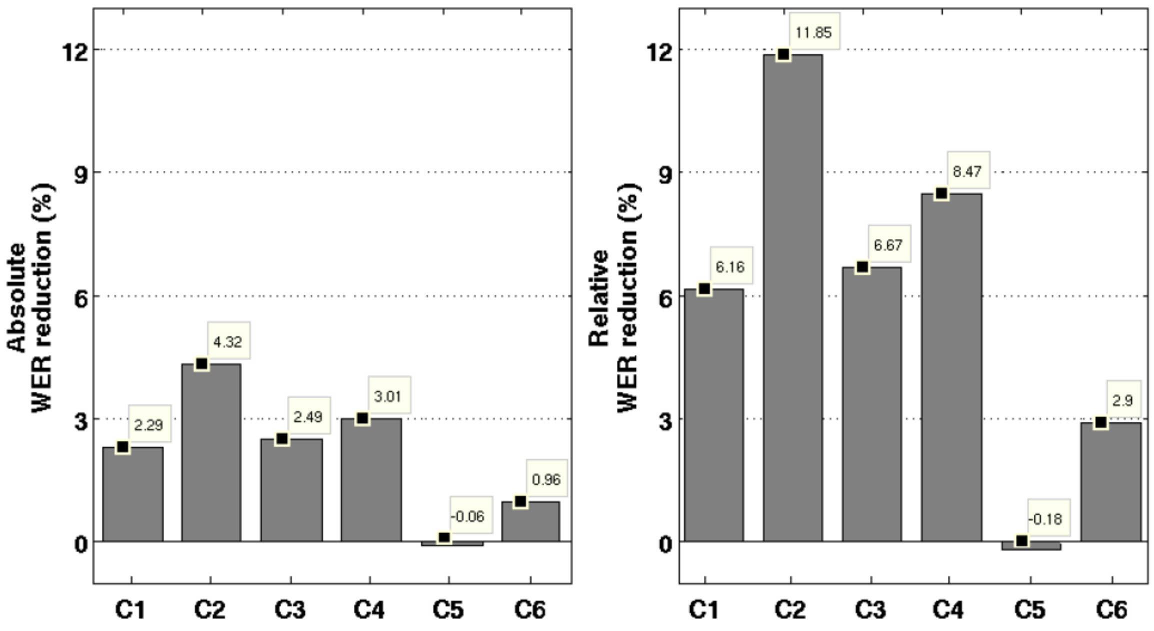


Figure 4.18: Entity-based voting algorithm WER reduction rates.

In order to illustrate the performance of the entity-based voting scheme, the WER reduction is plotted. Figure 4.18 reports the WER reduction (absolute and relative) achieved with our voting algorithm. The WER reduction reported in Figure 4.18 shows that our voting scheme is outperforming the original frequency of occurrence scheme in almost all the experiments. The relative WER reduction ranged from 3% to up to 12%. This reduction is considered impressive since ROVER performance has stagnated over the past few years and it has become more and more difficult to achieve even a small WER reduction. However, with the C5 binary combination, the entity-based voting algorithm performs slightly worse than the original ROVER algorithm. This can be explained by the fact that the C5 baseline WER is already the lowest which means performance is at its best. In other words, when the WER is low and decoders are highly optimized, the entity-based voting performs roughly the same as the original scheme.

To wrap up, in this experiment, a voting algorithm based on the entity detection algorithm has been proposed to improve the performance of the ROVER procedure. The voting scheme relies on entities in order to select the winner token at each slot of the WTN. Tests showed that this new approach outperforms the original voting scheme and achieve substantial WER reduction in most of the setups. Future research can focus on enhancing our proposed algorithm in a number of ways. One possible direction is to expand it to handle more than pairs of decoders. Indeed, combining the output of three or more decoders in our system should improve the results compared to using only two decoders. We could also look at improving the technique used to break ties. The random decision our system makes at present is not optimal, and some kind of intelligent decision process should give better results.

4.4 Summary

In this chapter, two of the applications introduced in Section 3.8 were studied through experimental tests. The first application was the query topic identification which was the main focus of the chapter. The problem was divided into three stages and a comprehensive study followed examining many parameters involved in the algorithm structure. At the first stage, the flexibility of entity detection and base topic identification towards noise was measured using synthesized data. The second stage provided an iterative method which tried to solve a real-world problem using KDD CUP 2005 datasets for query categorization. Finally, the stage three presented a full semantic analysis algorithm to solve KDD CUP 2005 problem using the full methodology described in this research. Many factors of the algorithm were tested in order to create the optimal solution which produced favorable results.

The second application was speech recognition problem which used a variation of the entity detection algorithm to create a voting mechanism for combining results of multiple decoders to generate a better transcription. The tests showed great improvements over baseline data which have not been improved upon for a long time.

Chapter 5
Conclusion and Future Research

5.1 Summary of the Study

This dissertation presented a methodology for semantic analysis of textual input based on an encyclopedic graph-based knowledgebase source. The presented methodology provided various algorithms and also provided a number of sample problems which can be tackled using the provided algorithms. It tried to show how the generic algorithm templates can be tailored for different types of applications by presenting the potential solutions for the presented sample problems. In each solution, some variation of the semantic analysis algorithm was used and justifications as why the algorithm was set up in a certain way were provided in each case. Finally, a couple of the applications were studied further using empirical experiments and tests. The numerical results of the performance of the algorithm in each setup was presented and compared to baseline methods. The experiments also tested various parameters of the algorithms and showed how changes in these parameters affect the final results.

The first part of the study was the introduction of Wikipedia and DBpedia as the knowledgebase used in the research. Wikipedia was introduced as a valuable source of semantic information. The main components of Wikipedia used to create the knowledgebase were individually explained. These components were: titles, infoboxes, wikilinks, and categories. All these components were the crucial blocks in creating the semantic graph. However, due to the human-readable structure of Wikipedia, there were challenges in converting this encyclopedia into a machine-readable format. Consequently, a semantic web version of Wikipedia was introduced as DBpedia which provided all the crucial components in computer-friendly linked data format. DBpedia is a collection of datasets which are compatible to RDF format since they use triples of “subject, predicate, object” and also use URIs to refer to items in their datasets. Such a structure can easily be transferred to a database for machine use. Moreover, the triple format and linked data nature of these dataset can easily be converted to a semantic graph.

The next part of the study introduced the semantic graph generated from Wikipedia or DBpedia. The semantic graph is comprised of two sub-graphs: entity graph and topic graph. This study heavily relies on the two concepts of entity and topic. The entity graph and the topic graph are thus the knowledge sources for each of these concepts. The entities were introduced as all possible knowledge items, such as people, places, events, organizations, and so on. The topics, on the other hand, are considered as a meta-layer of semantic knowledge which gives the entities structure and hierarchy. In other words, topics provide the classification of the knowledgebase and entities are the instances that belong to one or more of these classes. This means that the entity graph is connected to the topic graph and the connection was called the entity-topic connection layer.

Once the semantic graph is described, the study presented the first element of semantic analysis algorithms: entity detection. It was noted that any semantic analysis would require the entities to be first identified. The entities, therefore, are acting as entry points to the semantic graph unless an application is specifically working on topics and provides topics as the input. Most applications, however, would need to perform entity detection first. The proposed algorithm had included five divisions: processing, search, scoring, equivalency, and production. Each of these divisions was introduced, explained, and a formal mathematical definition for them was presented at the end. In summary, the processing division divided the input into blocks ready for further processing. Each block was called an input component. Search division was tasked to find entities matching each input component and called them candidate entities. Scoring division would assign a score to each candidate entity based on how close it matches the input component. Equivalency division tried to find entities that are equal to the candidate entities but are not found in search division and add them to candidate entity list. Finally, production division filtered out those candidate entities which scored lower than an acceptable threshold and returned the remaining high scored entities for further semantic analysis.

The second element of the semantic analysis was the introduction of semantic distance and commonality score. Since the main part of analysis is done using the semantic distance and commonality score, throughout this dissertation they are called as the semantic analysis themselves. This means while entity detection is a very important step in the analysis but the main analysis is performed when the proper semantic distances and the final scores using commonality score are calculated. The semantic distance was introduced as having various types. In one aspect, a semantic distance can be defined based on its source and destination e.g. whether it is between entities and entities, entities and topics, or topics and topics. Another aspect in defining the semantic distance was the direction which we observed could be upward, downward, strictly upward, or strictly downward. Some algorithms were presented on how to calculate each of these types of semantic distances. Finally, the commonality score stated that not only the semantic distance is important in calculating semantic relevancy but the number of different paths and how many entities or topics they have in common is also another aspect of semantic closeness. The commonality score was defined using a generic template equation which, in short, provided an aggregate function over the scores of candidate entities divided by their semantic distances to the goal targets. Using the entity detection, semantic distance, and commonality score, three basic semantic analysis methods were introduced next: entity analysis, topic analysis, and progressive analysis. Entity and topic analysis each examined how the analysis would be performed for each type separately and progressive analysis showed how a tracking and aging mechanism can be applied to the semantic analysis in general.

Once the description of semantic analysis methodology was completed, the study provided a number of real-world problems and applications which can benefit from the presented methodology. In each case the problem definition was explained and its important aspects were discussed. Afterwards, a version of semantic analysis algorithm was presented as the potential solution to that problem. It should be noted that for these applications, there were many other possible approaches available and even the presented potential solutions could be tweaked in many different ways. The goal was, however, to show at least one possible solution so the practicality of the methodology is demonstrated.

In order to further prove that the proposed semantic analysis can be implemented in real-world problems, the solutions for two of the applications were, in fact, implemented and tested using real-world datasets. The first application was query topic identification which was developed in three stages. First, it was only entity detection and calculation of immediate topics called base topics in order to examine the flexibility of the algorithm to noise words using synthesized datasets. The second stage used iterative algorithm for finding target topics instead of semantic distance and commonality. It used real datasets from KDD CUP 2005 competitions and scored favorably. Finally, the last stage used the full semantic analysis algorithm over the same data as the second stage. Many parameters of the algorithm were tweaked to study their effects on final results. An optimal system was created as the result of studying the parameters which scored highly in identifying new datasets from both KDD CUP 2005 and TREC 2007 sources. The second application was the speech recognition problem which used the entity detection algorithm to decide between the outputs of multiple speech decoder systems to achieve a final transcription of audio data. The experimental results showed impressive improvements over baseline methods.

5.2 Conclusion

The methodology presented in this dissertation has been shown to have a number of strong attributes to make it a valuable framework for various semantic analysis problems. The choice of a knowledge source such as Wikipedia which is an ever-growing up-to-date live knowledgebase makes it possible to provide more accurate analysis as more information is being added to the encyclopedia. If the implementation of the proposed algorithm uses the DBpedia online sources, there would be no need to update the knowledgebase as it will be automatically up-to-date since it receives the data directly from DBpedia network.

The number of different algorithms for semantic analysis provided in this study demonstrates the flexibility of the proposed methodology and its framework. The provided methods can be applied to a variety of applications. The number of application shown to use the methodology is an evidence of the flexibility of the methodology and framework.

Another attractive notion about the presented framework is the speed of the algorithms in various components. The implementation of this framework, specifically in Parla system and topic identification, was able to perform in real-time. The reason is that most of the algorithms in this study have linear complexity in each step. For instance, the search division of the entity detection algorithm has $O(N)$ time complexity where N is the number of the input words. This is done by the full-text indexing of the database containing entities which by using hash tables and various indices only requires $O(1)$ for each input word to return a list of candidate entities. Graph traversal algorithms have also at most $O(V)$ time complexity where V is the number of nodes in either the entity graph, the topic graph, or both. The reason for that is the condition that no node should be visited twice and, therefore, only a subset of nodes will ever be visited and only once for each. This means, in worst case scenario, the complexity is at most linear.

Finally, the experimental results presented here show the practicality of the methodology. In the query topic identification task, a ranking and classification algorithm was presented to find the best set of goal topics given user-specified keywords. To demonstrate its efficiency, a query classification system using this methodology was implemented. A thorough study of the algorithm was performed, focusing on each design decision individually and considering the practical impact of different alternatives. It was shown that our system's classification results compare favorably to those of the KDD CUP 2005 competition: it would have ranked 2nd on precision with a performance 10% better than the competition mean, and 7th in the competition on F1 also with a performance 10% better than the competition mean. Lastly, two blind tests on different datasets that were not used to develop the system were presented to validate the results. The consistency of high performance over a range of datasets demonstrates that the algorithms used are based on sound methodology and if applied correctly on real-world applications, they can provide valuable solutions.

In conclusion, the presented framework is shown to be versatile valuable work in semantic analysis which can be recreated and used by any researcher as all the resources are publicly available. The framework will always be recent as its data source is a live knowledgebase. It was shown to be very flexible and match numerous applications. Finally, it was shown to be practical and provide high performance results. Therefore, this methodology is proved to be a valuable tool for further semantic analysis by researchers in the field.

5.3 Future Work

The variety of the applications that can use the presented methodology provides a good platform for future research work in this area. There are, however, two important tasks which can immediately follow the work presented here. First, the speech recognition application, which was implemented and had produced improved results over baseline methods, only used the entity detection algorithm. The next step in this application is to add the progressive entity analysis. Since the algorithm is run over a number of transcription sets all related to a single large audio file, using progressive entity analysis make it possible to track the context. As it happens, same words keep repeating in a conversation and tracking the context makes it possible to have a history of found entities and reuse them in following transcription sets.

The second future work to do is the implementation of text prediction algorithm. This application again uses a progressive entity analysis algorithm. Any text document can act as the input for this application and, therefore, heavy testing of the algorithm is easily possible. It is a good idea to test all possible design parameters to find an optimal system for this application and then compare it to the other available methods.

Besides the above tasks, there exists a significant extension to this study which can be taken on in future research. As mentioned in challenges sections in previous chapters, there are certain limitations to this framework. The main issue is since there are no weights on the edges in the semantic graph, if an application needs single candidate entity for the analysis, this framework might not be able to provide the most accurate results as it operates mainly on sets of candidate entities. To remedy this limitation, one can continue this work by trying to calculate proper weights for the edges of the graph through learning methods or statistical approaches. Once done, the framework algorithms should be modified such that they incorporate weights into their traversal tasks.

Last but not least, the work on topic identification might be of interest to anyone developing query classification systems, text classification systems, or most other kinds of classification software. By using Wikipedia, a classification system gains the ability to classify queries into a set of almost 740,000 (and growing) categories covering most of human knowledge and which can easily be mapped to a simpler application-specific set of categories when needed, as it was done in this study. And while multiple alternatives at every design stage of the query topic identification system were considered and tested, it is possible to conceive of further alternatives that could be implemented on the same framework and compared to results presented here. Future work can focus on exploring these alternatives and further improving the quality of the classification. In that respect, one of the first directions to work in will be to integrate an automated corrector into the system, to address the problem of unknown words which was the source of 5.9% of the KDD CUP 800 queries that remained unclassified with our system.

Appendix A
Category Mappings

This appendix lists how we mapped the 67 KDD CUP categories to 99 corresponding Wikipedia categories in the September 2008 version of the encyclopedia.

KDD CUP Category	Wikipedia Category
Computers\Hardware	Computer hardware
Computers\Internet & Intranet	Internet
	Computer networks
Computers\Mobile Computing	Mobile computers
Computers\Multimedia	Multimedia
Computers\Networks & Telecommunication	Networks
	Telecommunications
Computers\Security	Computer security
Computers\Software	Software
Computers\Other	Computing
Entertainment\Celebrities	Celebrities
Entertainment\Games & Toys	Games
	Toys
Entertainment\Humor & Fun	Humor
Entertainment\Movies	Film
Entertainment\Music	Music
Entertainment\Pictures & Photos	Photographs
Entertainment\Radio	Radio
Entertainment\TV	Television
Entertainment\Other	Entertainment
Information\Arts & Humanities	Arts
	Humanities
Information\Companies & Industries	Companies
	Industries
Information\Science & Technology	Science
	Technology
Information\Education	Education
Information\Law & Politics	Law
	Politics
Information\Local & Regional	Regions
	Municipalities
	Local government
Information\References & Libraries	Reference
	Libraries
Information\Other	Information
Living\Book & Magazine	Books
	Magazines
Living\Car & Garage	Automobiles
	Garages
Living\Career & Jobs	Employment
Living\Dating & Relationships	Dating
	Intimate relationships
Living\Family & Kids	Family
	Children
Living\Fashion & Apparel	Fashion
	Clothing
Living\Finance & Investment	Finance
	Investment
Living\Food & Cooking	Food and drink
	Cooking
Living\Furnishing & Houseware	Decorative arts
	Furnishings
	Home appliances
Living\Gifts & Collectables	Giving
	Collecting
Living\Health & Fitness	Health
	Exercise
Living\Landscaping & Gardening	Landscape
	Gardening
Living\Pets & Animals	Pets
	Animals
Living\Real Estate	Real estate
Living\Religion & Belief	Religion

	Belief
Living\Tools & Hardware	Tools
	Hardware (mechanical)
Living\Travel & Vacation	Travel
	Holidays
Living\Other	Personal life
Online Community\Chat & Instant Messaging	On-line chat
	Instant messaging
Online Community\Forums & Groups	Internet forums
Online Community\Homepages	Websites
Online Community\People Search	Internet personalities
Online Community\Personal Services	Online social networking
Online Community\Other	Virtual communities
	Internet culture
Shopping\Auctions & Bids	Auctions and trading
Shopping\Stores & Products	Retail
	Product management
Shopping\Buying Guides & Researching	Consumer behaviour
	Consumer protection
Shopping\Lease & Rent	Renting
Shopping\Bargains & Discounts	Sales promotion
	Bargaining theory
Shopping\Other	Distribution, retailing, and wholesaling
Sports\American Football	American football
Sports\Auto Racing	Auto racing
Sports\Baseball	Baseball
Sports\Basketball	Basketball
Sports\Hockey	Hockey
Sports\News & Scores	Sports media
Sports\Schedules & Tickets	Sport events
	Seasons
Sports\Soccer	Football (soccer)
Sports\Tennis	Tennis
Sports\Olympic Games	Olympics
Sports\Outdoor Recreations	Outdoor recreation
Sports\Other	Sports

Appendix B
Publications

Journal Papers

- Milad AlemZadeh, Richard Khoury, and Fakhri Karray, “Query Classification using Wikipedia’s Category Graph”, *Journal of Emerging Technologies in Web Intelligence - Special Issue on Web Data mining*, Vol. 4, No. 3, pp 207-220, August 2012.
- Fakhri Karray, Milad Alemzadeh, Jamil Abou Saleh, and Mo Nours Arab, “Human-Computer Interaction: Overview on State of the Art”, *International Journal on Smart Sensing and Intelligent Systems*, Vol. 1, No. 1, pp. 137-159, March 2008.

Conference Papers

- Milad AlemZadeh, Kacem Abida, Richard Khoury, and Fakhri Karray, “Enhancement of the ROVER's Voting Scheme using Pattern Matching”, *The Third International Conference on Autonomous and Intelligent Systems, AIS'12*, Aveiro, Portugal, June 2012.
- Milad Alemzadeh, Richard Khoury, and Fakhri Karray, “Exploring Wikipedia’s Category Graph for Query Classification”, *The Second International Conference on Autonomous and Intelligent Systems, AIS'11*, pp. 222-230, Burnaby, BC, Canada, June 2011.
- Milad Alemzadeh and Fakhri Karray, “An Efficient Method for Tagging a Query with Category Labels using Wikipedia towards Enhancing Search Engine Results”, *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, IEEE Computer Society*, pp. 192-195, Toronto, Canada, September 2010.

Bibliography

[1] T. Berners-Lee, J. Hendler, and O. Lassila. (2001, May Issue) The Semantic Web. *Scientific American*. Available: http://www.scientificamerican.com/article.cfm?id=the-semantic-web

[2] TED, "Tim Berners-Lee on the next Web," ed, 2009.

[3] T. Berners-Lee. (2006). *Linked Data—Design Issues*. Available: http://www.w3.org/DesignIssues/LinkedData.html

[4] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data - The Story So Far," *International Journal on Semantic Web and Information Systems,* vol. 5, pp. 1-22, 2009.

[5] T. Berners-Lee. (2008). *Linked Open Data*. Available: http://www.w3.org/2008/Talks/0617-lod-tbl/

[6] *Wikipedia: The Free Encyclopedia*. Available: http://www.wikipedia.org/

[7] *DBPedia*. Available: http://dbpedia.org

[8] World Wide Web Consortium (W3C). (2004). *Resource Description Framework (RDF)*. Available: http://www.w3.org/RDF/

[9] World Wide Web Consortium (W3C). *Semantic Web Activity*. Available: http://www.w3.org/2001/sw/

[10] T. Berners-Lee. (2002). *The Semantic Web slides*. Available: http://www.w3.org/2002/Talks/04-sweb/Overview-1.html

[11] I. Herman. (2007). *Rule Interchange Format (RIF) Highlight*. Available: http://www.w3.org/2007/Talks/0507-rif/#(1)

[12] D. Fensel, H. Lausen, A. Polleres, J. d. Bruijn, M. Stollberg, D. Roman, *et al.*, *Enabling Semantic Web Services: The Web Service Modeling Ontology*, First ed.: Springer-Verlag New York, Inc., 2006.

[13] *Extensible Markup Language (XML)*. Available: http://www.w3.org/XML/

[14] E. Dumbill. (2000) The Semantic Web: A Primer. *OReilly XML.com*.

[15] M. Klein, "XML, RDF, and relatives," *Intelligent Systems, IEEE,* vol. 16, pp. 26-28, 2001.

[16] D. Gasevic, D. Djuric, and V. Devedzic, *Model Driven Architecture and Ontology Development*, 1st ed.: Springer, 2006.

[17] World Wide Web Consortium (W3C). (1997). *Resource Description Framework (RDF) Model and Syntax*. Available: http://www.w3.org/TR/WD-rdf-syntax-971002/

[18] F. Manola and E. Miller, *RDF Primer*: World Wide Web Consortium (W3C) Recommendation, 2004.

[19] World Wide Web Consortium (W3C). *Semantic Web Interest Group*. Available: http://www.w3.org/2001/sw/interest/

[20] World Wide Web Consortium (W3C). *RDF Interest Group*. Available: http://www.w3.org/RDF/Interest/

[21] World Wide Web Consortium (W3C). *RDF Current Status*. Available: http://www.w3.org/standards/techs/rdf#w3c_all

[22] J. Heflin, J. Hendler, and S. Luke, "SHOE: A Knowledge Representation Language for Internet Applications," *Technical report,* 1999.

[23] P. D. Karp, V. K. Chaudhri, and J. Thomere, "XOL: An XML-Based Ontology Exchange Language," *Technical Report: SRI International,* 1999.

[24] World Wide Web Consortium (W3C). (2004). *RDF Vocabulary Description Language 1.0: RDF Schema*. Available: http://www.w3.org/TR/rdf-schema/

[25] J. Hendler and D. L. McGuinness, "The DARPA agent markup language," *IEEE Intelligent systems*, vol. 15, pp. 67-73, 2000.

[26] D. Fensel, F. Van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider, "OIL: An ontology infrastructure for the semantic web," *Intelligent Systems, IEEE*, vol. 16, pp. 38-45, 2001.

[27] I. Horrocks, "DAML+OIL: A Description Logic for the Semantic Web," *IEEE Data Engineering Bulletin*, vol. 25, pp. 4-9, 2002.

[28] D. L. McGuinness and F. Van Harmelen, "OWL web ontology language overview," *W3C recommendation*, vol. 10, pp. 2004-03, 2004.

[29] W3C OWL Working Group, "OWL 2 Web Ontology Language: Document Overview," *W3C Recommendation*, October 2009.

[30] J. Cardoso, "The semantic web vision: Where are we?," *Intelligent Systems, IEEE*, vol. 22, pp. 84-88, 2007.

[31] Stanford University. *The Protégé Ontology Editor and Knowledge Acquisition System*. Available: http://protege.stanford.edu/

[32] A. Kalyanpur, B. Parsia, E. Sirin, B. C. Grau, and J. Hendler, "Swoop: A web ontology editing browser," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 4, pp. 144-153, 2006.

[33] *SWOOP: Semantic Web Ontology Editor*. Available: http://code.google.com/p/swoop/

[34] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke, "OntoEdit: Collaborative ontology development for the semantic web," *The Semantic Web—ISWC 2002*, pp. 221-235, 2002.

[35] L. Ding, P. Kolari, Z. Ding, and S. Avancha, "Using ontologies in the semantic web: A survey," *Ontologies*, pp. 79-113, 2007.

[36] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: implementing the semantic web recommendations," presented at the Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, New York, NY, USA, 2004.

[37] *Apache Jena*. Available: http://jena.apache.org/

[38] World Wide Web Consortium (W3C). (2008). *SPARQL Query Language for RDF*. Available: http://www.w3.org/TR/rdf-sparql-query/

[39] V. Haarslev and R. Moller, "Description of the RACER System and its Applications," in *Proceedings of the 2001 International Workshop on Description Logics (DL-2001)*, Madrid, 2001.

[40] Racer Systems GmbH & Co. KG. *RacerPro*. Available: http://www.racer-systems.com/

[41] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, pp. 51-53, 2007.

[42] Clark & Parsia. *Pellet: OWL 2 Reasoner for Java*. Available: http://clarkparsia.com/pellet

[43] World Wide Web Consortium (W3C). *Linked Data*. Available: http://www.w3.org/standards/semanticweb/data

[44] W3C SWEO Community Project. *Linking Open Data*. Available: http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[45] *GeoNames*. Available: http://www.geonames.org/

[46] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," presented at the Proceedings of the 16th international conference on World Wide Web, Banff, Alberta, Canada, 2007.

[47] F. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: A Large Ontology from Wikipedia and WordNet," *Web Semantics,* vol. 6, pp. 203-217, 2008.
[48] R. Cyganiak and A. Jentzsch. *Linking Open Data cloud diagram*. Available: http://lod-cloud.net/
[49] Creative Commons. *Attribution 3.0 Unported (CC BY 3.0)*. Available: http://creativecommons.org/licenses/by/3.0/
[50] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer, "Semantic Wikipedia," in *Proceedings of the 15th international conference on World Wide Web*, Edinburgh, Scotland, 2006, pp. 585-594.
[51] *Semantic MediaWiki*. Available: http://semantic-mediawiki.org/
[52] D. Vrandecic and M. Krötzsch, "Semantic MediaWiki," *Semantic Knowledge Management Integrating Ontology Management, Knowledge Discovery, and Human Language Technologies,* pp. 171-179, 2008.
[53] Y. Li, Z. Zheng, and H. Dai, "KDD CUP-2005 report: facing a great challenge," *SIGKDD Explor. Newsl.,* vol. 7, pp. 91-99, 2005.
[54] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin*, et al.*, "Q^2C@UST: our winning solution to query classification in KDDCUP 2005," *ACM SIGKDD Explorations Newsletter,* vol. 7, pp. 100-110, 2005.
[55] D. Shen, J. T. Sun, Q. Yang, and Z. Chen, "Building bridges for web query classification," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 131-138.
[56] D. Shen, R. Pan, J. T. Sun, J. J. Pan, K. Wu, J. Yin*, et al.*, "Query enrichment for web-query classification," *ACM Transactions on Information Systems (TOIS),* vol. 24, pp. 320-352, 2006.
[57] H. Cao, D. H. Hu, D. Shen, D. Jiang, J. T. Sun, E. Chen*, et al.*, "Context-aware query classification," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 3-10.
[58] J. Fu, J. Xu, and K. Jia, "Domain Ontology Based Automatic Question Answering," presented at the Proceedings of the 2009 International Conference on Computer Engineering and Technology - Volume 02, 2009.
[59] J. Yu and N. Ye, "Automatic Web Query Classification Using Large Unlabeled Web Pages," in *Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management*, 2008, pp. 211-215.
[60] S. M. Beitzel, E. C. Jensen, D. D. Lewis, A. Chowdhury, and O. Frieder, "Automatic classification of Web queries using very large unlabeled query logs," *ACM Transactions on Information Systems (TOIS),* vol. 25, p. 9, 2007.
[61] J. Hu, G. Wang, F. Lochovsky, J.-t. Sun, and Z. Chen, "Understanding user's query intent with wikipedia," in *Proceedings of the 18th international conference on World wide web*, Madrid, Spain, 2009, pp. 471-480.
[62] R. Khoury, "Query classification using Wikipedia," *IJIIDS,* vol. 5, pp. 143-163, 2011.
[63] R. Khoury, "Exploiting Wikipedia in query understanding systems," in *Digital Information Management (ICDIM), 2010 Fifth International Conference on*, 2010, pp. 292-297.
[64] P. Schönhofen, "Identifying document topics using the Wikipedia category network," *Web Intelli. and Agent Sys.,* vol. 7, pp. 195-207, 2009.
[65] Z. Minier, Z. Bodo, and L. Csato, "Wikipedia-Based Kernels for Text Categorization," in *Proceedings of the Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2007, pp. 157-164.

[66] P. Wang, J. Hu, H.-J. Zeng, and Z. Chen, "Using Wikipedia knowledge to improve text classification," *Knowl. Inf. Syst.*, vol. 19, pp. 265-281, 2009.

[67] S. Banerjee, K. Ramanathan, and A. Gupta, "Clustering short texts using wikipedia," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, The Netherlands, 2007, pp. 787-788.

[68] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives, "DBpedia: A Nucleus for a Web of Open Data," in *In 6th International Semantic Web Conference, Busan, Korea*, 2007, pp. 11-15.

[69] Wikimedia Foundation. *Wikipedia Statistics*. Available: http://stats.wikimedia.org/EN/

[70] *Freebase - Wikipedia Article*. Available: http://en.wikipedia.org/wiki/Freebase_(database)

[71] *Wikimedia Downloads*. Available: http://dumps.wikimedia.org/

[72] M. Krötzsch, D. Vrandecic, and M. Völkel, "Wikipedia and the Semantic Web - The Missing Links," ed, 2005.

[73] D. Aumueller and S. Auer, "Towards a Semantic Wiki Experience - Desktop Integration and Interactivity in WikSAR," presented at the Proceedings of 1st Workshop on The Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure, Galway, Ireland, Nov. 6th, 2005.

[74] S. Campanini, P. Castagna, and R. Tazzoli, "Platypus Wiki: a Semantic Wiki Wiki Web," in *1st Italian Semantic Web Workshop*, 2004.

[75] R. Tazzoli, P. Castagna, and S. E. Campanini, "Towards a semantic wiki wiki web," 2004.

[76] A. Souzis, "Building a Semantic Wiki," *IEEE Intelligent Systems*, vol. 20, pp. 87-91, 2005.

[77] A. Pipitone and R. Pirrone, "A framework for automatic semantic annotation of Wikipedia articles," presented at the 6th Workshop on Semantic Web Applications and Perspectives, Bressanone, Italy, 2010.

[78] S. Auer and J. Lehmann, "What Have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content," in *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria*, ed, 2007, pp. 503-517.

[79] F. Wu and D. S. Weld, "Autonomously semantifying wikipedia," presented at the Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, Lisbon, Portugal, 2007.

[80] F. Wu and D. S. Weld, "Automatically refining the wikipedia infobox ontology," presented at the Proceedings of the 17th international conference on World Wide Web, Beijing, China, 2008.

[81] D. S. Weld, F. Wu, E. Adar, S. Amershi, J. Fogarty, R. Hoffmann, *et al.*, "Intelligence in wikipedia," presented at the Proceedings of the 23rd national conference on Artificial intelligence - Volume 3, Chicago, Illinois, 2008.

[82] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, *et al.*, "DBpedia - A crystallization point for the Web of Data," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, pp. 154-165, 2009.

[83] M. Mohamed, L. Jens, A. ren, S. Claus, and H. Sebastian, "DBpedia and the live extraction of structured data from Wikipedia," *Program: electronic library and information systems*, vol. 46, pp. 157-181, 2012.

[84] *DBpedia Downloads Page*. Available: http://wiki.dbpedia.org/Downloads

[85] *DBpedia Datasets Page*. Available: http://wiki.dbpedia.org/Datasets

[86] C. Bizer, "DBpedia 3.7 released, including 15 localized Editions," in *DBpedia*, ed, 2011.

[87] A. Miles, B. Matthews, M. Wilson, and D. Brickley, "SKOS core: simple knowledge organisation for the web," presented at the Proceedings of the 2005 international conference on Dublin Core and metadata applications: vocabularies in practice, Madrid, Spain, 2005.

[88] A. Miles and S. Bechhofer, *SKOS Simple Knowledge Organization System Namespace Document - HTML Variant*: World Wide Web Consortium (W3C) Recommendation, 2009.

[89] *DBpedia 3.7 Downloads Page*. Available: http://wiki.dbpedia.org/Downloads37

[90] J. Sonin. (2007). *Wikipedia Concept Map*. Available: http://www.flickr.com/photos/juhansonin/407874864/

[91] *Wikipedia:Categorization*. Available: http://en.wikipedia.org/wiki/Wikipedia:Categorization

[92] D. Milne and I. H. Witten, "Learning to link with wikipedia," presented at the Proceedings of the 17th ACM conference on Information and knowledge management, Napa Valley, California, USA, 2008.

[93] O. Medelyan, I. Witten, and D. Milne, "Topic Indexing with Wikipedia," presented at the Proceedings of the first AAAI Workshop on Wikipedia and Artificial Intelligence, 2008.

[94] D. Milne and I. Witten, "An effective, low-cost measure of semantic relatedness obtained from Wikipedia links," in *AAAI 2008*, Chicago, IL, 2008.

[95] *WikipediaMiner*. Available: http://wikipedia-miner.cms.waikato.ac.nz/help/

[96] Z. Syed, T. Finin, and A. Joshi, "Wikipedia as an Ontology for Describing Documents," presented at the Proceedings of the Second International Conference on Weblogs and Social Media, 2008.

[97] Z. Syed, T. Finin, V. Mulwad, and A. Joshi, "Exploiting a Web of Semantic Data for Interpreting Tables," in *Proceedings of the Second Web Science Conference*, 2010.

[98] Z. Syed and T. Finin, "Unsupervised techniques for discovering ontology elements from Wikipedia article links," presented at the Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading, Los Angeles, California, 2010.

[99] Z. Syed, "Wikitology: a novel hybrid knowledge base derived from wikipedia," University of Maryland at Baltimore County, 2010.

[100] B. J. Jansen, A. Spink, and T. Saracevic, "Real life, real users, and real needs: a study and analysis of user queries on the web," *Inf. Process. Manage.*, vol. 36, pp. 207-227, 2000.

[101] M. Lee, B. Pincombe, and M. Welsh, "An Empirical Evaluation of Models of Text Document Similarity," in *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, ed: Erlbaum, 2005, pp. 1254-1259.

[102] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, pp. 391-407, 1990.

[103] S. T. Dumais, "Latent semantic analysis," *Annual Review of Information Science and Technology*, vol. 38, pp. 188-230, 2004.

[104] *SwiftKey*. Available: http://www.swiftkey.net/

[105] K. Coursey and R. Mihalcea, "Topic identification using Wikipedia graph centrality," presented at the Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers, Boulder, Colorado, 2009.

[106] H. T. Dang, D. Kelly, and J. Lin, "Overview of the TREC 2007 Question Answering Track," in *NIST Special Publication: SP 500-274 The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings*, 2007.

[107] B. Cao, J.-T. Sun, E. W. Xiang, D. H. Hu, Q. Yang, and Z. Chen, "PQC: personalized query classification," presented at the Proceedings of the 18th ACM conference on Information and knowledge management, Hong Kong, China, 2009.

[108] K. Abida and F. Karray, "Systems combination in large vocabulary continuous speech recognition," in *Autonomous and Intelligent Systems (AIS), 2010 International Conference on*, 2010, pp. 1-6.

[109] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER)," in *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, 1997, pp. 347-354.

[110] H. Jiang, "Confidence measures for speech recognition: A survey," *Speech Communication,* vol. 45, pp. 455-470, 04// 2005.

[111] J. Fiscus, J. Garofolo, M. Przybocki, W. Fisher, and D. Pallett, "1997 English Broadcast News Speech (HUB4)," 1998.

[112] D. Huggins-Daines, "CMU Sphinx Open Source Models," 2008.

[113] D. Graff and C. Cieri, "English Gigaword," 2003.