Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

NJIT New Jersey's Science & Technology University

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

ABSTRACT

USING AN ONTOLOGY TO IMPROVE THE WEB SEARCH EXPERIENCE

by
Tian Tian

The search terms that a user passes to a search engine are often ambiguous, referring to homonyms. The results in these cases are a mixture of links to documents that contain different meanings of the search terms. Current search engines provide suggested query completions in a dropdown list. However, such lists are not well organized, mixing completions for different meanings. In addition, the suggested search phrases are not discriminating enough. Moreover, current search engines often return an unexpected number of results. Zero hits are naturally undesirable, while too many hits are likely to be overwhelming and of low precision.

This dissertation work aims at providing a better Web search experience for the users by addressing the above described problems.To improve the search for homonyms, suggested completions are well organized and visually separated. In addition, this approach supports the use of negative terms to disambiguate the suggested completions in the list. The dissertation presents an algorithm to generate the suggested search completion terms using an ontology and new ways of displaying homonymous search results. These algorithms have been implemented in the Ontology-Supported Web Search (OSWS) System for “famous people.”

This dissertation presents a method for dynamically building the necessary ontology of “famous people” based on mining the suggested completions of a search engine. This is combined with data from DBpedia. To enhance the OSWS ontology,

Facebook is used as a secondary data source. Information from “people public pages” is mined and Facebook attributes are cleaned up and mapped to the OSWS ontology.

To control the size of the result sets returned by the search engines, this dissertation demonstrates a query rewriting method for generating alternative query strings and implements a model for predicting the number of search engine hits for each alternative query string, based on the English language frequencies of the words in the search terms. Evaluation experiments of the hit count prediction model are presented for three major search engines. The dissertation also discusses and quantifies how far the Google, Yahoo! and Bing search engines diverge from monotonic behavior, considering negative and positive search terms separately.

USING AN ONTOLOGY TO IMPROVE THE WEB SEARCH EXPERIENCE

by
Tian Tian

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

January 2012

Copyright © 2012 by Tian Tian

ALL RIGHTS RESERVED

APPROVAL PAGE

USING AN ONTOLOGY TO IMPROVE THE WEB SEARCH EXPERIENCE

Tian Tian

Dr. James Geller, Dissertation Advisor (date)
Professor of Computer Science, NJIT

Dr. Narain Gehani, Committee Member (date)
Professor and Dean of Computer Science, NJIT

Dr. Dimitrios Theodoratos, Committee Member (date)
Associate Professor of Computer Science, NJIT

Dr. Michael Halper, Committee Member (date)
Professor of Information Systems, NJIT

Dr. Soon Ae Chun, Committee Member (date)
Associate Professor of Information Systems, CUNY

Dr. Hayato Yamana, Committee Member (date)
Professor of Computer Science and Engineering, Waseda University, Japan

BIOGRAPHICAL SKETCH

Author: Tian Tian

Degree: Doctor of Philosophy

Date: January 2012

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2012

- Bachelor of Science in Computer Science,
  Beijing University of Posts and Telecommunications, Tianjin, P. R. China, 2007

Major: Computer Science

Presentations and Publications:

T. Tian, J. Geller, S.A. Chun, "A Prediction Model for Web Search Hit Counts Using Word Frequencies," Journal of Information Science, vol. 37, issue 5, pp. 462-475, Sage Publishing Co., 2011.

C. Ochs, T. Tian, J. Geller, S.A. Chun, "Google Knows Who is Famous Today: Building an Ontology from Search Engine Knowledge and DBpedia," 5th IEEE International Conference on Semantic Computing (ICSC), Palo Alto, CA, 2011.

T. Tian, J. Geller, S.A. Chun, "Enhancing Interface for Ontology-Supported Homonym Search," CAiSe'11 Workshop on Semantic Web Search (SSW), London, UK, 2011. Lecture Notes in Business Information Processing (LNBIP), issue 83, pp. 544-553, Springer Verlag, Berlin, 2011.

T. Tian, J. Geller, S.A. Chun, "Improving Web Search Results for Homonyms by Suggesting Completions from an Ontology," 2nd ICWE Workshop on Semantic Web Information Management (SWIM), Vienna, Austria, 2010. Lecture Notes in Computer Science (LNCS), issue 6385, pp. 175-186, Springer, 2010.

T. Tian, J. Geller, S.A. Chun, "Predicting Web Search Hit Counts," 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp.162-166, Toronto, Canada, 2010.

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. James Geller, who not only served as my research advisor, providing valuable and countless resources, insight, and intuition, but also constantly gave me support, encouragement, and reassurance. I am heartily grateful as well to my co-advisor, Dr. Soon Ae Chun, who gave me inspiration and thoughtful guidance throughout this research. Special thanks are given to Dr. Narain Gehani, Dr. Dimitrios Theodoratos, Dr. Michael Halper and Dr. Hayato Yamana for actively participating in my committee.

I also would like to thank Christopher Ochs, Yuwen Sun and Shrutee Shah for their assistance during the course of my study. Finally, I would like to express my deep gratitude to all my family members for their love and support.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

1.1 Problems

Users' information needs in the digital era can be fulfilled by keyword-based search engines. Such search engines have become the universal catalogs for world-wide resources. Unlike the old library catalogs that are mostly searchable by fixed fields (e.g., by authors, titles, and keywords predefined by authors), modern Web search engines provide a flexible, easy way to express search terms. Users' Web searches have never become easier without the search engines.

However, the results returned by the search engines cannot always satisfy the Web users. Covering world-wide resources on the Web, the search engines often return millions of pages for one search, which may lead to information overload [1]. To determine which documents are useful, users often have to sift through many hits to find a few that are relevant [1], or repeatedly refine their search terms.

This dissertation work aims at providing a better, faster and easier search experience for the users. The ideal search results would be less overwhelming, and yet contain more relevant hits. In this dissertation, two approaches are discussed to improving the users' Web search experience.

1.1.1 Problem 1: Irrelevant Search Results (Especially for Homonyms) Returned by the Search Engines

In what is shaping up to be the "Century of the Web" a computer literate person with an information need is likely to eschew traditional sources of information such as libraries, yellow pages and newspapers and turn immediately to a Web search engine. Such information needs define the work sphere ("from where can I source this industrial part that I need") as much as private life ("where is a nice, affordable restaurant near my home") and everything in between ("I need a cheap flight for a job/private trip"). Thus, the quality of the *search experience* of a user has become of major importance. A user wants an answer, and she wants it **now,** and she wants it many times a day. Search engines are expected to provide correct results quickly, and with a minimal amount of user interaction.

To satisfy this expectation of an agreeable search experience, major efforts have gone into improving both the backends and frontends of common search engines. For example, Google has switched from making users type in complete search terms and hitting return (or clicking a button) to suggesting to the user what she is mostly likely to ask for. Such suggested completions [2] have also been introduced by other search engines. Figure 1.1 shows Google's suggested completions for the query term "Barack Obama." Google has access to the search terms entered by its millions of users, which makes it easy for them to propose crowd-based suggested completions.

Changes to the backend are harder to discern for the user, but search results are often long lists of snippets referring to a few relevant links among many irrelevant results

[1, 3]. Previous research has focused on refining the search terms and on filtering the results, to improve the precision of the returned snippets [1, 4, 5].

Figure 1.1 Google's suggested completions for search term "Barack Obama."

Search engines also suffer from three common problems in Natural Language Processing, the synonym problem, the homonym problem, and the wrong granularity problem. The synonym problem appears in the form that the user might send a different term to the search engine than what is contained in a document that would provide a relevant answer. Thus, a query term "43rd president of the US" might miss documents with George Bush, even though these two terms are synonymous.

The wrong granularity problem would appear when a user performs a search with a general or wide term, and a relevant document contains only a more specific or narrow term (or vice versa). Thus, a search for "government officials having been impeached" might not bring up President Clinton, who was indeed impeached.

The third problem in this category occurs when a search term is a homonym [6, 7, 8] (a term with multiple meanings or multiple referents) and the user does not know that. For example, when using the search term "President George Bush" without any further

qualification, it might refer to George W. Bush or his father George H. W. Bush, the 43rd and 41st president of the United States, respectively. If the user wants information about the former, she would get results about both of them with this search term, which is an unintended and misleading result.

Thus, when using a search engine to satisfy an information need about a homonymous concept, a user is faced with two kinds of problems. She might get an overwhelming number of responses about one homonym, especially if this meaning is more popular, while the second homonym with a less popular meaning that she might be really interested in is hidden in a snippet on a much later page of hits, returned by the search engine. This is the case with lopsided preferences in meanings. For instance, the "Michael Jackson" who is a singer is much more popular than the basketball player of the same name. Hence many more search results contain references to the singer. In this situation, the user is at least aware that the results she is getting are not about the basketball player that she has been looking for. When formulating the initial query, it escaped her attention that there are two concepts for her search term and that more information might be available on the Web about the homonym that she is not interested in.

The situation is even worse if the user is completely unaware of the fact that the search term is a homonym with two (or more) references, and all results that appear on the first few pages of hits are to the "wrong" reference. For example, a user located in the New York area, who types "Penn Station" into Google will see many references to Penn Station in New York City (NYC) and some references to Penn Station in Newark. These two Penn Stations are separated by a 20 minute train ride. Unbeknownst to her, there is

also a Penn Station in Philadelphia, Pennsylvania. However, a reference to the latter does not appear on the first page of search results.

1.1.2 Problem 2: Undesirable Size of Results Returned by the Search Engines

Common search engines often return too many results for an initial query, which may lead to information overload [1, 9]. Most initial searches result in thousands or even millions of relevant Web pages available for the given search terms. Such a result might be perceived as overwhelming [10]. A user is not impressed by a million hits. Very often she wants only a few hits that are all highly relevant to the search that was performed and that address her immediate information need. While search engines have improved to the point that the desired answer is often on the first page of results, users still may have to sift through many hits to find a few that are relevant [1], or repeatedly refine their search terms. In the latter case there is a danger of overspecifying the search, e.g., by using long phrases in double quotes, with the effect that no results at all are returned.

It has been reported that search engines normally stop at about the 1000th result, with all other matching pages remaining hidden from users [11]. Besides, research results have shown that search engine users often give up their search after the first try, examining no more than 10 documents or the first page of hits [12]. Eye-tracking studies showed that we can expect clicks only for the top few results, and that the search engine will probably receive almost no feedback about any result ranked above 100 [13]. A user study by iProspect also showed that 62% of search engine users don't look past the first page of results [14]. Only 10% of users click on results beyond the third page [14]. To get

useful results without sifting through pages of hits, users often have to resort to a "feedback loop" of repeated queries with increasingly refined search terms.

To provide a scenario for the problem we are addressing, if a user attempts to find information about the US Senator Paul Simon, as opposed to the singer Paul Simon, she will get pages of results about the singer, with the desired results about the senator hidden among those. To find information about the senator, she will need to repeatedly refine her search terms by adding words associated with politics. Another technique to increase the number of useful results in the first few pages (i.e., the precision) is to include negative search words. Thus, a negative search word of the form ''–singer'' should reduce the number of irrelevant results.

However, this query refinement approach has its own problems. If a user specifies too many positive or negative search words, relevant hits could be excluded, i.e., the recall would suffer. It would be especially undesirable if no page hits at all are returned. The interplay between the user and the browser could be described as a feedback loop. The long range goal of this research is to automate this feedback loop in a manner that is invisible to the user and implement it as a plug-in. The browser with this plug-in would process the search terms of the user but would not actually show the results to her if there are too many hits. The user would also never know when a search was attempted that resulted in zero page hits.

1.2 Solutions

The goals of this research are:

- To categorize suggested completions by the different meanings of homonyms and present them to the user in an improved way reflecting those different meanings;
- To control the result size of the search engine results by predicting the search hit counts and adding additional search terms.

1.2.1 Solution for Improving the Suggested Search Completions

The goal in this part of research is to improve the user search experience with suggested search completions in three ways. First, the display of suggested search term completions should be categorized visually to make it clear that homonymous terms exist. For this, knowledge of the classes that terms belong to is necessary. This is the kind of knowledge normally contained in ontologies.

Secondly, the knowledge in the ontologies should be used to increase the precision of results, by making the suggested completions as discriminating as possible. One tool for making Web searches more focused is to use negative search terms in addition to the normal "positive" search terms. Naturally, the suggested search completions should not be over-specified to the point that the search engine would not return any results. As the public does not have access to the "most common search terms" collected by commercial search engines, they cannot be used to generate suggested completions. Instead ontologies are used both for creating the suggested completions and for providing the knowledge needed to visually categorize them.

Including negative search terms in the search queries is a powerful tool for discriminating between wanted and unwanted results. In the past, negative search terms have not been used in suggested completions. This dissertation discusses the generation of suggested completions with negative search terms and hint at the problems that arise out of this pursuit.

The ontology has been extended to enrich the information provided for the search terms (see Chapter 3).

Thirdly, combining support for the homonymous search terms, a Web search mechanism is developed and implemented with an improved search experience for the user that minimizes the necessity for input actions. This dissertation presents the "vertical view" mechanism (see Section 2.3.1) and discusses the new instant feature incorporated into the Web search system (see Section 2.3.2).

1.2.2 Solution for Building the Ontology for the Suggested Completions

As discussed in Section 1.2.1, the goal of this work is to provide better suggested completions to users, by disambiguating homonyms and appending suggested terms from a robust ontology. Ontologies were chosen to serve this purpose, because they are well suited for defining the important notions (classes, relations, objects) of a domain, using concepts, roles, and instances (individuals), as they are known in Description Logics [15].

An ontology was developed, containing basic knowledge of more than 5000 musicians and more than 3000 basketball players, whose information is extracted from Wikipedia. The ontology has been submitted to the Ontology Design Patterns (ODP) as an exemplary ontology.

This dissertation addressed the crucial question of how to enhance the system's ontology. The goal is to improve the ontology in four ways. Firstly, a method is presented to mine the suggested completions from a search engine. Secondly additional information is extracted from DBpedia [16] (see Chapter 3). Thirdly, this dissertation describes the process of expanding the ontology dynamically during the normal operation of the OSWS System. Finally, it discusses the process of enhancing the ontology by mining Facebook as a secondary resource (see Chapter 4).

1.2.3 Solution for Predicting the Search Hit Counts

As addressed in Section 1.1.2, the users often need to repeatedly refine their search terms in the query to get more relevant results from the search engines, which results in a feedback loop. One approach is a query rewriting method (also query expansion) that the browser would utilize to reduce the number of hits by appending additional words to the search that are in line with the interests of the user. The previous research used an approach similar to relevance feedback, however based on an ontology, to provide additional search words [17, 18]. Fu et al. [19], Navigli and Velardi [20] and Andreou [21] have presented various methods and algorithms to expand queries by applying ontologies. The query rewriting mechanism augments user search terms with positive words from an ontology. The specific model of query rewriting consists of adding additional terms to the user query. For example, the query ‘‘Michael Jackson’’ can be augmented by additional terms such as ‘‘singer,’’ ‘‘king of pop,’’ ‘‘thriller,’’ etc. More details about the query rewriting method can be found in [17, 18, 22]. This model has been extended to negative search words.

Negative search words can be derived, e.g., from a user model of an individual user. This user model would contain subjects and their associated terms that the user is definitely not interested in. However, the expanded search criteria can result in a list of alternative search strings that need to be processed, one after another, by the user, until the result is satisfactory. To automate this manual feedback loop, the output of the query expansion approach can be processed by a browser plug-in 'in the background' and only results that would not overwhelm the user should be reported to her. The idea of running queries in the background is inspired by [23]. As part of a feedback loop, many such searches would have to be executed, which would result in an unacceptable waiting time. Thus, running several or many queries in the background is not practical.

Instead of executing searches in the background, this dissertation is therefore attempting to predict the hit count estimates that will be returned for different expanded search terms. Only a search for which the plug-in predicts a number of hits between pre-specified limits will be executed. The output will only be presented to the user if the prediction was correct, i.e., the number of results is between the pre-specified limits. These limits could, for example, be 10 and 100, with a certain error range permitted. Thus, one focus of this paper is on the prediction mechanism for alternative expanded search terms. Such a mechanism helps users to avoid 'zero results' as well as information overload from too many low precision results.

The major search engines return a list of hits, preceded by a number of approximately how many hits should be expected. This number has been referred to as 'hit count estimate'. It has been observed that the quality of hit count estimates goes down considerably when transitioning from one search word to two search words [24].

The hit count estimates of the search engines were used in this research, because real hit counts are difficult to obtain by manual counting, whenever there are many hits.

This dissertation presents an approach to developing a model for predicting the number of hits for different combinations of search words. To develop the hit count prediction model, a series of searches were conducted with search terms ranging from 1 – 5 words, correlation models were built between the search term frequencies and hit count estimates returned by the search engines. Different prediction models have been developed, based on the number of search words, allowing for up to five positive and up to five negative search words.

To validate the prediction model, a series of searches were conducted. Their hit count estimates reported by three commonly-used search engines, Google, Yahoo! and Bing were compared with the hit counts reported by the prediction model.

During these experiments, it is observed that the hit count estimates for many search words do not observe the monotonicity requirements expected as a minimal constraint; that is, whenever a positive or negative search word is added to a prior search, the number of hits should go down (monotonicity). Thus, the second part of this work analyzes this (mis)behavior for positive and negative search words. A failure of a negative search word to reduce the number of results should be considered more serious than a failure of a positive search word. The results indicate that monotonicity often does not hold, and that there are wide differences between search engines.

1.3 Related Work

This dissertation work aims at improving the suggested completions for homonymous names of famous people. There is other research trying to solve the problem of personal name disambiguation, but mostly in the context of clustering techniques [25, 26, 27].

Semantic search on the Web, which aims at enabling more intelligent Web searches, has become one of the hottest Semantic Web research topics [28]. Keyword-based approaches have been studied by many researchers in the field to improve the search process [28]. For example, [29] improves the traditional search method by augmenting the search results with relevant data aggregated from the Semantic Web. Falcons is a keyword-based search engine for concepts and objects on the Semantic Web [30]. SWSE [31] and Sig.Ma [32] allow users to locate RDF entities via keyword search [28]. Some of the mentioned studies have also addressed the problem of query disambiguation, considering user preferences or heuristics [28]. Chapter 2 discusses the approach to improving the query disambiguation, in order to improve the search experience.

Ontologies were used to provide the suggested completions in the search system. In order to build such ontologies, search engine knowledge is mined. Yossef et al. [33] have used the public interface to mine and sample the search engines' query logs for other research purposes. The ontology consists of the search engine knowledge as well as the data extracted from DBpedia [34]. DBpedia is a large multi-domain ontology, which has been commonly used for ontology building [35].

Besides DBpedia, Facebook was used as a secondary resource to mine knowledge about famous people. Over the past few years, Facebook has become the largest social

networking site. Millions of users have integrated Facebook into their daily practices [36]. Research has been done on mining data from social networks. For example, Thelwall et al. have mined MySpace comments to detect the emotion among them and to examine how they differ among users with different age and gender [37]. Chu et al. have mined Facebook live data concerning social networking forensics [38]. Xu et al. studied mining user opinions in social network services [39]. Numerous tools have been developed to mine social networks. For example, SONAR is an API for gathering and sharing social network information [40]. POLYPHONET was built as a social network extraction system [41].

This dissertation also discusses the approach to hit count prediction modelling. Adding words from an ontology to a user's search terms was demonstrated in the previous ontology-based search system [6] as a method for improving the precision of Web search results. Thelwall observed that search engine results are now widely used for measurement purposes by researchers in Webometrics [42], and for commercial activities such as Web analytics and search engine optimization. Cilibrasi and Vitanyi used search engine hit counts to measure word similarity [43]. Similar work in the Semantic Web community, using hit count estimates to calculate similarities between resources in a semantic network, can be found in [44, 45, 46]. Search engine hit counts were used to measure the popularity of a famous person [6] (see Chapter 3). Thus, there is a need for research into the reliability of the results of search engines [11]. Other research has focused on the consistency of the results of search engines. The hit count estimates that they report for queries are interesting for at least two reasons. Webometric research has used these hit counts as input for many studies of Web information, e.g., to determine

how many pages in one country link to another [11]. Secondly, it is useful to know how reliable the estimates reported by search engines are [11].

Due to their great commercial and technological success, search engines have been studied by many researchers. Yossef and Gurevich used random samples from a search engine's index to measure the size of the search engine [47]. However, the hit count estimates are not utilized in their work and there is no research on predicting the search engine hit count estimates. The query pool in [47] is built by crawling the ODB directory, while this dissertation research is based on the British National Corpus (BNC) [48], a 100 million word collection of samples of written and spoken British English. Moreover, word frequencies are not considered in [47]. Matsuo et al. have done researches to estimate the Google hit counts [49]. However, their method requires many actual Google queries to be sent to evaluate the co-occurrence of terms. (See Section 4.2 for the method of utilizing the co-occurrence of search terms). Thus, in order to estimate the hit count for one query, several other Google queries have to be made in Matsuo's method. Obviously, if this dissertation uses prediction in order to avoid spending time making the actual Google queries, Matsuo's method would not be suitable for the purpose of predicting hit counts in real time.

1.4 Structure of the Dissertation

Chapter 2 describes the approach to improving Web search experience for homonyms by suggesting completions from an ontology and enhancing the search interface. Chapter 3 presents the ontology used in Chapter 2 and the methodology to dynamically build and expand the ontology. Chapter 4 describes the approach to improve

the ontology presented in Chapter 3 by mining Facebook [50] as a secondary resource. Chapter 5 presents the approach to hit count prediction modeling. Chapter 6 is devoted to problems of search engines' handling of negative and positive search words. Chapter 7 concludes the dissertation work.

CHAPTER 2

IMPROVING WEB SEARCH RESULTS FOR HOMONYMS BY SUGGESTING COMPLETIONS FROM AN ONTOLOGY

2.1 Introduction

This chapter is based on work published in [6] and [51]. As mentioned in Section 1.1.1, current browsers don't deal well with search requests when the search terms are homonyms. To improve the users' search experience with the homonymous terms, this chapter describes the approach to improving the search results for homonyms by suggesting completions from an ontology.

In the previous research on an ontology-supported Web search system, the user was presented with a number of choices of additional search terms for her input. She could mark such terms as positive, i.e., they should be included in the Web search results, by clicking on associated check boxes (see Figure 2.1 and 17]). One problem with this approach was that users do not want to be bothered by (too many) questions. A more benign approach to eliciting additional information from a user can be seen in the use of suggested completions. While a user types in the first (few) word(s) of her search, the search engine displays up to ten suggested search completions, which will possibly describe the search that the user had in mind. These completions are presumably based on the observed frequencies of many searches of other search engine users [2]. While the user continues to type, the suggested completions change rapidly and are often limited to fewer than ten. Most major search engines have such a mechanism. Google calls them

"query suggestions" appearing in the "search box" [2], Yahoo calls them "search assistant" [52], and Bing calls them "search suggestions" [53].

Figure 2.1 Search screen example of previous Ontology-Supported Web Search System.

Another weakness of the approach in [17] was that it did not make use of the information that may be inferred by a form of closed-world assumption from the terms that the user did *not* select with a check mark. According to the documentation of major search engines, the use of negative search words, marked with a minus sign before the word(s), constitutes a particularly powerful tool for discriminating between different results. Thus, the approach in [17, 18, 22] is extended in this dissertation work by adding negative search terms.

Google's suggested completions have problems described in Section 1.1.1. They do not reflect distinctions between different concepts that are expressed by the same word or the same multi-word term (homonyms). Suggested completions also do not appear to be optimized for discrimination between homonyms. Appending well-chosen negative

search words to a search term given by the user would result in improved discrimination between homonyms of that search term, if the appended words are characteristic for one of the homonymic senses. However, the use of too many negative search terms might exclude relevant results, i.e., the recall would suffer. It would be especially undesirable if the search is so over-specified that no results are reported at all.

The suggested additional search terms in [17] (see Figure 2.1) were derived from an ontology. For a given user input, all homonymous concepts were located in the ontology. Then choices of additional terms were generated by looking at neighboring concepts in the ontology. Thus "Michael Jackson" is categorized as a singer, or in more technical terms, Michael Jackson is an instance of the class "singer." Thus, a statement of this nature with a check box was suggested to the user. If the user checks this box, then the word "singer" was automatically appended to the Web search query before executing it.

Finding information about Michael Jackson the singer on the Web is clearly not a problem. There are millions of hits for this search term. However, when somebody is interested in Michael Jackson the basketball player, or in any one of the other over 20 Michael Jacksons that have achieved some kind of fame over the years, then the task of finding relevant information becomes much more difficult. Appending negative search words such as "-songs" and "-lyrics" makes this task easier, by excluding the most widely used homonym of Michael Jackson the singer.

2.2 Generating Ontology-Based Search Term Completions

2.2.1 An Algorithm for Suggested Completions with Positive Terms Only

The basic idea of generating suggested completions with *positive* search terms was not changed from [17], however the interface model was changed considerably, improving both on the previous work and on common search engines. The following pseudocode demonstrates the processing steps.

ALGORITHM *DISPLAY_SEARCH_SUGGESTIONS*

INPUT: *SEARCH_TERM, KNOWLEDGE_BASE*

OUTPUT: *Display of SEARCH_SUGGESTIONS*

BEGIN

NODE_COLLECTION = {}

FOR EACH *NODE* **IN** *KNOWLEDGE_BASE*

IF *NODE* contains *SEARCH_TERM*

NODE_COLLECTION = *NODE_COLLECTION* U{*NODE*}

/* *NODE_COLLECTION* now contains all homonyms */

ITH_SUGGESTION = 1

IF *size_of(NODE_COLLECTION) > 4*

NODE_COLLECTION = MOST_COMMON(NODE_COLLECTION)

/* *NODE_COLLECTION* now contains at most 4 homonyms */

FOR EACH *NODE* **IN** *NODE_COLLECTION*

NEIGHBOR_LIST = {}

FOR *N* **IN** *NEIGHBORS_PLUS_GRANDPAR(NODE)*

/* We add one additional level in the IS-A hierarchy to the immediate neighbors. */

NEIGHBOR_LIST = NEIGHBOR_LIST U {<*REL, N*>}

/* Pairs of all neighbors and their connecting relationships are collected in a list. */

PRIOR_LIST = PRIORITIZE(NEIGHBOR_LIST)

/* Pairs with important relationships, such as IS-A are placed first in the list. */

SEARCH_SUGGESTIONS[ITH_SUGGESTION] = PRIOR_LIST

ITH_SUGGESTION++

SEARCH_SUGGESTIONS = LIMIT_SIZE(SEARCH_SUGGESTIONS)

/* At most 12 lines are displayed over all homonyms. */

DISPLAY_WITH_SEPARATORS(SEARCH_SUGGESTIONS)

/* Suggestions for each homonym are displayed, visually separated from each other. */

END

The algorithm *DISPLAY_SEARCH_SUGGESTIONS* uses the following sub-algorithms: *MOST_COMMON* returns at most four homonyms. The selection is done based on the number of hit counts for each homonym. These hit counts are recorded in the ontology during creation time. More details can be found in Section 2.2.2.

NEIGHBORS_PLUS_GRANDPAR returns for every instance in the ontology all neighboring nodes that are one link away from it, plus the "grand parent," i.e., the IS-A parent of the class it is an instance of.

PRIORITIZE sorts the list of neighbors by importance. The importance is determined by the types of connecting relationships. Thus IS-A relationships to parent classes are considered more important than lateral semantic relationships. See Section 2.2.2 for more details on the importance of different relationship types. If several neighbors are connected by the same relationship type, then the order of the connected concepts is chosen arbitrarily.

LIMIT_SIZE controls the total size of the output. In order to avoid overloading the user with information and in order to achieve a behavior similar to existing search engines, the total number of search suggestions displayed is limited to at most 12. The number 12 is divisible by 2, 3, and 4, which makes it a good choice for 2, 3, or 4 homonyms.

Finally, the sub-algorithm *DISPLAY_WITH_SEPARATORS* creates the actual dropdown box that is shown to the user. It contains the computed search suggestions with appropriate separators to express the semantic distances between them.

Altogether the algorithm specifies the following behavior. A user types words of a search term into the search box. The algorithm locates nodes (classes or instances) in the stored ontologies that correspond to the input words. If only one node is located, then there is no problem with homonymy, at least according to the knowledge incorporated in the set of all loaded ontologies. On the other hand, if two (or more) nodes are located in the ontologies that match the user input, then additional processing is necessary.

For each located node, its neighbors[1] in the ontology network are retrieved, starting with the parent(s), if it is a class, or if it is an instance, the class that it is an instance of. Neighbors that are common to more than one sense (meaning) of the search term are eliminated, as they have no discriminatory power. The algorithm now appends subsets of these retrieved terms to the user terms to generate several suggested completions.

Knowledge from different domains is assumed to be stored in separate ontologies. However, when using this implemented knowledge, all ontologies are considered connected and combined into a single knowledge base.

Consider the following abstract example.

- The user types in two words *A B*, for example *A*=Michael and *B*=Jackson.
- The system identifies two concepts referred to as *A B*, let us call them *AB1* and *AB2*.
- *AB1* is an instance of *K*. *AB1* has a neighbor *L*.
- *AB2* is an instance of *M*. *AB2* has a neighbor *N*. The concepts *K*, *L*, *M* and *N* are distinct.
- The search engine generates the following suggested completions, three for *AB1* and three for *AB2*:
- *A B K*;
- *A B L*;
- *A B K L*;
- *A B M*;
- *A B N*;

[1] The immediate neighbors of a class are the following: parent classes (more general), child classes (more specific) and classes that are reachable from it by traversing a "semantic relationship." The immediate neighbors of an instance are the class which the instance belongs to, and the object properties and the data type properties of the instance.

- and *A B M N*.

- The total number of suggested completions is limited by a threshold and controlled by strict priorities in which order to select neighbors (Section 2.2.2).

- The suggested completions are presented to the user in a way that visually separates the *AB1* meaning from the *AB2* meaning, for example by using a bold line to separate them or by different background colors (see Section 2.2.2).

2.2.2 The Ontology-Supported Web Search (OSWS) System

A special-purpose search engine (also known as vertical search engine) limits its coverage in order to tailor the search results to one well-defined application domain [54]. There are many special-purpose search engines on the Web, e.g., Google Image Search, Yahoo Video Search, Twitter Search, Technorati, etc. Our goal is to build a special-purpose search engine for the domain of famous people.

The Ontology-Supported Web Search (OSWS) System for "famous people" provides search suggestions based on the user input, every time she types a new character. As seen in Figure 2.2, after the user completes the search term "Martina," the system finds all the famous people in the knowledge base with "Martina" in their names. (The display is updated after every single letter that the user types.) Additional background information about these famous people is extracted from the knowledge base for generating suggested completions. In this example, the tennis players Martina Hingis and Martina Navratilova and the singer Martina McBride are found. From the information related to these three famous people the suggested completions in the dropdown box are generated and displayed to the user.

Figure 2.2 Interface of the OSWS System for search term "Martina."

In more detail, for each concept of a famous person of the same name, all immediate neighbors along with the connecting relationships are retrieved from the ontologies. In the OSWS System, the first proposed suggestion about a famous person is always based on the class (modeling the occupation) of the person, which defines the name of the domain that the person belongs to. For instance, Martina Hingis has the first suggested completion "Martina Hingis tennis player" and Martina McBride has the first suggested completion "Martina McBride singer."

Then the remaining suggestions about each famous person are constructed based on the knowledge retrieved from the ontologies. They may include the background information of a person like the date of birth and the place of birth, and sometimes the birth name. Besides, for musicians, the ontology stores the genres of music the artist performs. For sportsmen, the league and the team he or she belongs to are represented in the ontology. For instance, in Figure 2.2, from the suggested completions the user could learn that Martina McBride plays country music, adult contemporary music, and country pop music, which she may not have been aware of.

Note that the OSWS processing happens in real time, whenever a new letter is added to the search term. Thus, when the user finishes the word "Martina," these four Martinas in the ontology are candidates for completion. They should be viewed as "homonymous according to the input currently made available to the search engine."

Different famous Martinas are separated by horizontal lines and background colors. This separation clearly expresses the fact that there are conceptual distances among the homonyms expressed by different sets of suggested completions. This makes it easier for the user to learn or remember that she is dealing with a homonym. Current

search engines do not support such a separation. In fact, the visual display in the example expresses the fact that Martina Hingis is conceptually closer to Martina Navratilova (both tennis players) than to Martina McBride (the singer) by separating the tennis players by thin lines from each other, and by separating them with a heavy line from the singer.

Besides the separating lines, the background color design in the dropdown box also distinguishes famous people from different domains. In Figure 2.2, the suggestions for the two tennis players are generated by the system with a blue background, in contrast to the suggested completions of the singer that are displayed with a pink background. Four preselected background colors are used for the at most four homonyms for which suggested completions may be displayed.

After the user chooses one suggestion that fits her search needs and clicks the "Google Search" button, a result page will be generated, using the Google AJAX API. In the example shown in Figure 2.3, when the user types "Adam S" the visible choices switch to "Adam Sandler," "Adam Sherburne," and "Adam Schmitt," based on the people known in the ontology. As noted before, there are cases where complete names are still homonymous (recall examples of George Bush, Michael Jackson, etc. in Chapter 1).

Figure 2.3 Interface of the OSWS System for search term "Adam S."

To avoid overwhelming the user with too many suggestions, and to stay close to the Google look and feel of the interface, the system is designed to show up to a maximum of 12 suggestions for a maximum of four famous people.

As noted in Chapter 1, there may be too many potential suggested continuations for one concept, and a selection process is required. The selection of lines for one homonym is achieved by assigning different priorities to different relationship types. For example, the IS-A link to the domain name (occupation) is considered to have the highest priority. For musicians, the genres of music they play have higher priorities than their dates of birth and places of birth. For basketball players, the team and league they play in are treated as more important than their birth information. Thus, the system only shows the high priority suggestions if there is more knowledge in an ontology than available space in the dropdown box. For example, in Figure 2.4, it shows 12 suggestions in the search box by eliminating the date of birth and place of birth information of the singer Michael Jackson, since these have the lowest priorities.

Figure 2.4 Interface of the OSWS System for search term "Michael Jackson."

If there are more than four homonyms (such as the over 20 Michael Jacksons) then the OSWS System selects up to four senses based on some criteria. There are two candidate approaches for this selection process. One possible selection criterion is the amount of information available in the ontologies about each sense. Thus, senses with a large amount of attached knowledge should be preferred over other senses. This is based on the pragmatic assumption that system implementers would not make the effort of including a large amount of information about a concept in an ontology if that concept is considered unimportant. However, this selection approach requires mature ontologies covering many domains with rich knowledge. Unfortunately, such ontologies do not always exist, and it is still a big challenge to build them (Section 2.3).

In the absence of sufficiently complete ontologies a second approach is needed, which is the one used in the OSWS System. A possible criterion to select the most popular homonyms is by using the Google hit count estimates. It is assumed that people with higher hit count estimates are more popular and famous. For instance, the query "Michael Jackson singer" returns almost twice the number of Web pages than the query "Michael Jackson basketball." Thus, Michael Jackson the singer should be preferred over the others.

The suggested completions in the search box of the OSWS System change dynamically after every single input character, just as in Google. The response time of the OSWS System in the current implementation is near instantaneous, limited more by the typing speed of the user than by the response time of the system. The initial ontologies in the system contained semantic information about more than 5000 musicians, more than 3000 basketball players and a sampling of sportsmen in other domains. Altogether, only

three of the over 20 Michael Jacksons known to us are included in the ontologies. Clearly, with more and much larger ontologies in the OSWS knowledge base, the response time will degrade. More ontology building and experimentation is needed to investigate the effect of ontology size on response time.

2.2.3 Suggested Completions with Positive and Negative Search Words

The previous work [17, 18, 22] did not use negative search terms. The idea of using negative search terms is akin to mutual inhibition as it occurs in neural networks. If different neurons compete for achieving maximum activation, they inhibit neighboring neurons. This should be seen only as a metaphor, not as a technical model, as there are vast differences between the numeric approach of a neural network and the symbolic approach of an ontology. Based on this metaphor, if the user types in Michael Jackson and the ontology knows about Michael Jackson the singer and Michael Jackson the basketball player, then two useful suggested completions would be:

Michael Jackson Singer –Basketball

Michael Jackson Basketball –Singer

In both those suggested completions, a bold font is used to indicate the words that have been entered by the user. Thus, neighbors of a node that are used as positive search terms for one homonym should be introduced as negative search terms for the other homonym. As of writing, none of the major existing search engines suggests completions with negative search terms.

Figure 2.5 Use of negative terms in the OSWS System for search term "Michael Jackson."

Many search engine users appear to be unfamiliar with the meaning of a minus sign (–) in front of a search word. Thus, suggesting a completion with a minus sign is syntactically unsatisfactory. Rather, the above completions need to appear as:

Michael Jackson Basketball [*but not*] Singer

Michael Jackson Singer [*but not*] Basketball

Figure 2.5 shows the use of negative search terms in the OSWS System for search term "Michael Jackson."

Using negative search words in suggested completions raises both conceptual and practical problems. One big practical problem that has been discovered in this research was that three major search engines do not process negative search terms as would be expected from their documentations or from a logical understanding of the meaning of "negative" words. This problem will be discussed in Chapter 6.

Conceptually, there is an issue of how many negative terms should be added in each proposed query. Adding too many might result in getting no Web search results at all. It is highly undesirable to propose a continuation to a user which will then not result in any Web page hits. As a "straw man" solution for this latter problem, one could imagine that all of the proposed complete search terms (user terms plus continuation terms) are passed on to the underlying search engine "in the background." The results are then "reviewed" by the Ontology-Supported Web Search System. Only if a complete search term results in a desirable (positive) number of Web hits would this search term be proposed to the user.

The reason why running several Web searches in the background is a straw man but not a practical solution is that a single search would take an unacceptable amount of time. Proposed continuations are presented to the user while she is typing and must be generated very quickly to avoid any appreciable delay. This difficulty has engendered the interest in methods for the high speed prediction of the number of Web sites that are likely to be returned for a given search term. Results of this research are presented in Chapter 5.

2.3 Enhancing the Interface for Ontology-Supported Homonym Search

Google has rolled out the new instant feature (Google Instant) in September 2010 [55]. As the user starts to type the first few letters of her search term, Google Instant automatically shows snippets of results for the most popular search term (the first suggested completion) that begins with those letters. The snippets appear below the box with suggested completions. As the user keeps typing, the snippets are dynamically updated. The user does not need to press enter or click the search button.

Google Instant helps the user to get better search results faster. Most importantly, seeing results as the user types her input helps her formulate a better search term by providing her with instant feedback [55]. However, Google Instant only displays result snippets for the first suggested completion in the drop-down suggestion box, even if the user moves the mouse over other suggestions. There are cases in which the user may be interested in making a choice between two suggested completions below the first one, but she cannot get instant feedback about them by using Google Instant. Rather she has to "make a commitment" to one of the suggested completions by clicking on it, which

defeats the purpose of the instant feature, which is to minimize the number of user actions necessary to obtain a satisfactory result.

The following subsections discuss methods for combining Ontology-Supported Web Search with techniques for providing the user with an improved search experience. First, to help the user choose the desired homonym, the result screen is split vertically and show result page snippets for the different homonymous terms next to each other. This gives the user more detailed information about different categories to help her decide what results best fit her interests. This display method gives the different homonymous concepts the same visibility "in the first row" and thus does not have a vertical bias towards one of the senses of the user's search term.

Secondly, the system shows the result page snippets for one specific homonym every time the user hovers with the mouse on top of a suggested completion for a selectable time period, currently set to two seconds or longer. This helps the user to acquire a deeper and clearer understanding of the suggested term without having to make a choice. Google, at this point in time, does not automatically change the displayed snippets when the mouse is moved down to a lower suggested completion.

2.3.1 Improving the OSWS System with Parallel Result Display for Homonyms

One innovative feature we have implemented into the OSWS System is the "vertical view" of the returned search results. As the suggestions were categorized for the different homonyms of a search term, such as singers, sports players, politicians, etc., the display screen below the suggested completions box is also divided into a few (two to four) vertical panels with result snippets for the different homonyms, so that the panels are

displayed next to each other. Every vertical panel contains the Google results for the first suggested completion of a different homonym. In this way, none of the homonyms is given the privileged position of being displayed in the first row.

As seen in Figure 2.6, after the user types "Adam," four homonymous famous "Adams," the singers Adam Lambert, Adam Sandler, Adam Bomb and the Basketball player Adam Morrison are found in the ontology and suggested to the user. Before she decides which Adam fits her interest or moves her mouse to a lower suggested completion, the system instantly shows the result snippets for the first suggestion of each of the four different Adams. The display screen is tiled into four vertical panels, which are respectively the results for the search terms "Adam Lambert singer," "Adam Sandler singer," "Adam Bomb singer" and "Adam Morrison basketball player." Thus, the top down order in the suggested completions is reflected in the left-to-right order of the tiled windows containing result snippets.

As mentioned before, due to the structure of the ontology, the first suggested completion of each homonym is in most cases the occupation of the famous person. The returned snippets give the user richer and more detailed information about different homonyms to help her decide what result best fits her interests.

The OSWS System shows the parallel results for all the homonyms only at the beginning of the search. During the time that the user types letters into the search box, the system features the vertical view of the returned results to help the user make a choice among the homonyms. Once the user moves the mouse down, the display is changed to the instant feature view that we are about to introduce in Section 2.3.2.

Figure 2.6 Interface of the OSWS System with the parallel result display for search term "Adam."

2.3.2 Improving the OSWS System with Instant Visual Feedback

This section describes an improved implementation of the instant feature [55]. As mentioned in Section 2.3.1, as soon as the user types letters into the search box, the system will automatically display the result snippets for the first suggested completion of each of the homonyms, if there are any, next to each other. Once the user moves the mouse to another suggestion in the drop-down menu and hovers over it, the OSWS System changes the display from horizontally parallel panels to one single panel, which contains the returned snippets of the suggested completion the user is apparently interested in because she is hovering on top of it.

As opposed to Google, not only the snippets of the first suggestion will be displayed, but if the user moves down to any other suggested completion, the snippet display is dynamically refreshed and updated. Thus, the user will instantly see results elaborating the suggested completions, to help her acquire a deeper and clearer understanding of the meaning of the suggested completion she is on. This helps her improve her searches significantly and efficiently. Google currently only shows instant results for the first suggested completion.

To avoid an overload of the server and to better react to the user's interests, the system has set a criterion for the minimum time required to be spent by the user hovering over a selected completion. Thus, only if the user stays for a while on one suggested completion (without clicking), the corresponding result snippets are displayed. The hover time is currently set to two seconds, but this is a user-adjustable parameter (see Figure 2.7). To make it possible for the user to move back and forth between different suggested completions, snippets are cached locally and do not have to be reloaded from the server.

Once the user clicks on one of the suggested completions, the display will be changed. The selected suggestion will be shown in its entirety in the search box. By clicking the search button next to it, the user will be led to the regular Google result page of the selected search term. For the extraction of the Web information we are using the Google AJAX API.

Page options are also added the in the OSWS System to display the suggested completions of the less popular homonyms (beyond the top 4). Note that in Figure 2.8, the numbers to the right show 5-8. This indicates the homonymous terms now showing are ranked 5th to 8th in popularity. The blue left arrow below the search box leads to the suggestions of the more popular homonyms, while the blue right arrow shows the suggestions of the less popular homonyms.

Figure 2.7 Adjustable hover time of the instant feature in the OSWS System.

Figure 2.8 Page options of the suggestions in the OSWS System.

CHAPTER 3

BUILDING THE "FAMOUS PEOPLE" ONTOLOGY FROM SEARCH ENGINE KNOWLEDGE AND DBPEDIA

3.1 Ontology Unde Venis?

(Ontology, from where do you come?) Probably the biggest problem with all ontology-based systems is from where to take the necessary ontologies, which is the same with the OSWS System (Chapter 2). Developing the ontologies in-house is time consuming and person-hour and/or budget intensive. Wide-scale ontology reuse has still not materialized, even though the Semantic Web [56], ontology search engines such as Swoogle [57] and ontology repositories [58, 59] have attempted to solve this problem. Many approaches to automatically generating or extending ontologies [17] have met with partial success but have also not reached the state of "shrink wrapped solutions."

Outside of Medical Informatics, where huge terminological repositories are the norm, many ontologies are small. There appears to be little interest in building large, fact-oriented, regularly structured ontologies. Researchers prefer to focus on intricately structured, abstract "upper level ontologies" [60, 61] or on rules and axioms. Building even one ontology with sufficient depth, breadth and domain coverage is a major challenge. Building ontologies just for the many existing categories of famous people in science, religion, art, history, politics, etc., and their many subcategories such as biology, chemistry, physics, computer science etc. in science, is a daunting task, even if we limit those ontologies mostly to simply-structured facts and instance categorizations.

Nevertheless, human intelligence relies on both rules and large numbers of simply-structured facts, including basic categorizations. For example, humans know about a large number of people how they are categorized, such that each one may be categorized as a

- family member
- friend
- workmate
- politician
- sportsman or sports woman
- "somebody from the history book"
- service provider (electrician, plumber, …)
- teacher, student or classmate
- etc., etc.

or as completely unknown. Therefore, ontologies have been developed, consisting of information on US musicians and athletes. These ontologies were constructed by programmatically extracting data, such as genres for musicians and leagues for athletes, from Wikipedia.

This chapter is based on work published in [6] and [62].

3.2 The Original "Famous People" Ontology

As mentioned in Section 3.1, the knowledge represented in these ontologies was mined from Wikipedia using a Web parsing program and stored in a temporary relational database. Not all information was available about every one of the over 5000 musicians.

The representation of three homonymous examples, "Michael Jacksons," in the knowledge base can be seen in Figure 3.1.

Classes in the figure are represented as boxes. Instances are shown as ellipses. IS-A links are drawn as arrows from the child class to the parent class. Dashed arrows connect instances to the classes that they are instances of. Finally, lines terminated by little black squares indicate semantic relationships other than IS-A and instance of relationships.

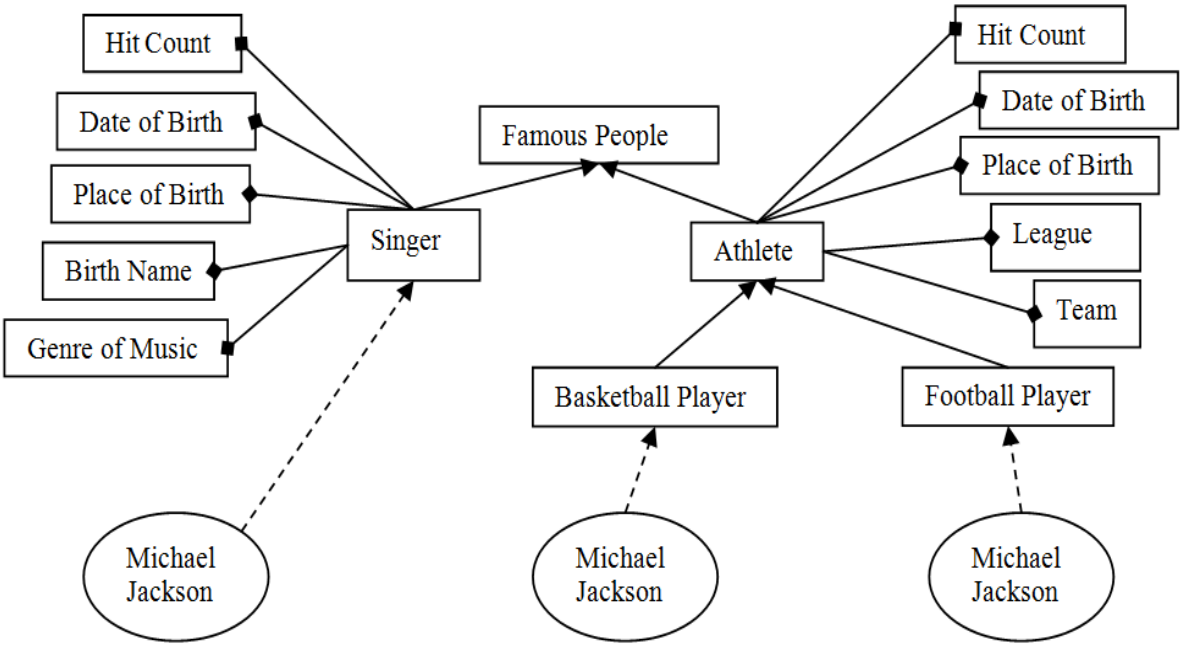

Figure 3.1 Excerpt from the "famous people" knowledge base with homonym example "Michael Jackson."

The Google hit count estimates have been collected and assigned to the appropriate instances of famous people while building the musician and basketball player ontologies. Thus, this information is available before the user starts with her search. However, this solution has several disadvantages. Hit counts are not stable. For example, after the singer Michael Jackson's untimely death, the number of hits greatly increased. Thus the previously mentioned ontology-size-based criterion would be preferable.

A Protégé ontology was built, using the Protégé API, by extracting the mined data from the temporary database. One of the problems encountered in this process was the uniqueness requirement of Protégé. Thus, the city "Washington" and the state "Washington" could not both be represented by the same atom, and we had to append qualifiers to distinguish between them, in this case by appending the letters "DC" to the city.

To make this work available to the ontology community, the musician ontology (see Figure 3.2) has been submitted to Ontology Design Patterns (ODP) as an exemplary ontology: http://ontologydesignpatterns.org/wiki/Ontology:Musician_Ontology.

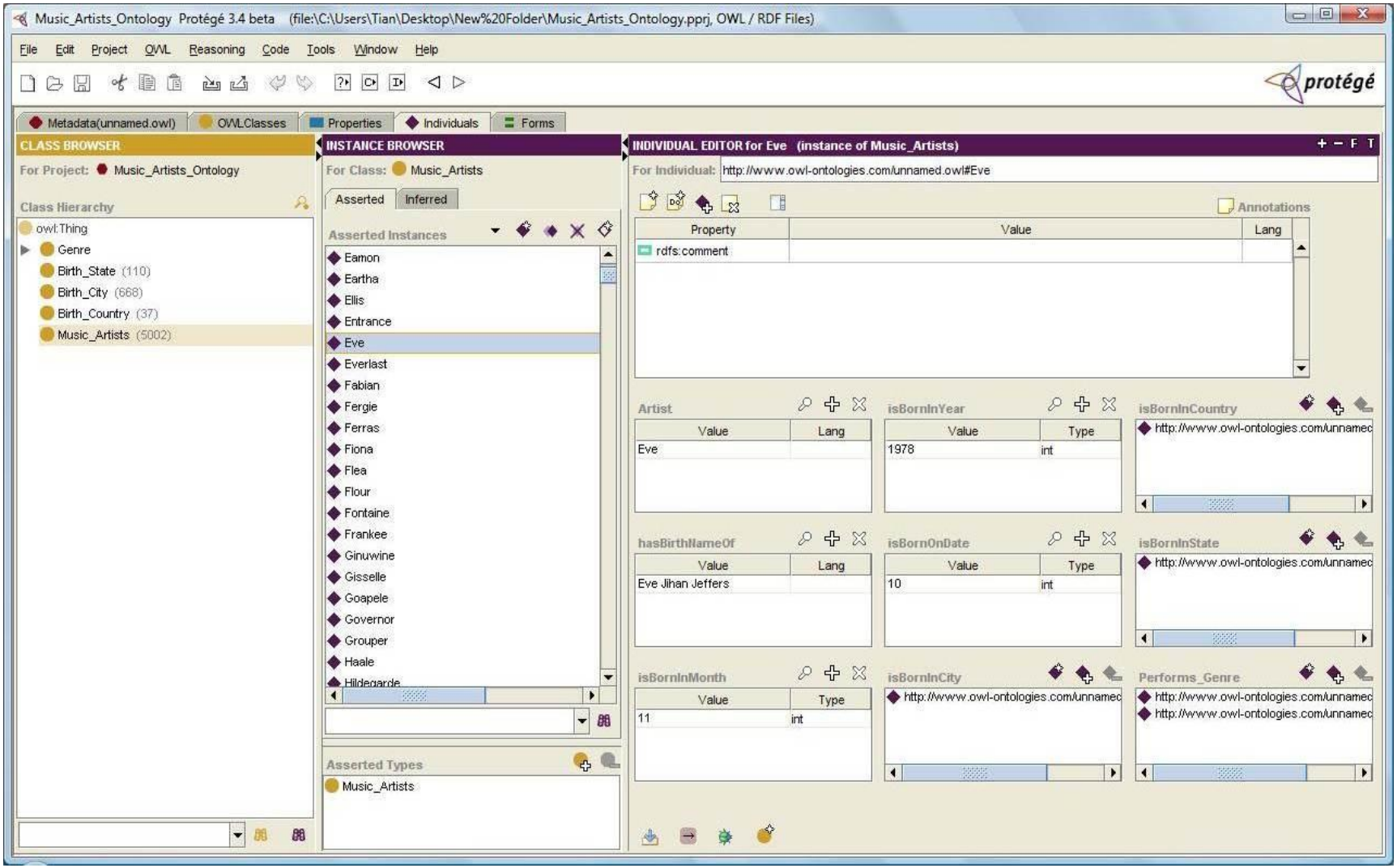

Figure 3.2 Instance example of the musician ontology in Protégé.

One shortcoming of the OSWS System was the limited scope of knowledge stored in its ontology. By focusing only on American musicians, basketball players, American football players, and a few selected others the ontology was of limited practical use. Additionally, the types of information stored in the ontology were generally not helpful for an end-user. Many instances in the ontology were limited to information about their dates of birth, given names, and locations where they were born. While most instances also included information on musical genre(s) or athletic league, respectively, users would prefer a wider variety of class-specific information, such as current sports team, released albums, movies starred in, etc.

The remaining of this chapter describes work on a principled method for building an extended ontology of famous people. The new ontology covers over 3,200 famous persons including artists, athletes, politicians, scientists, and others (see Section 3.3). Moreover, the OSWS ontology is dynamically expanded during use (see Section 3.4).

3.3 Building the New "Famous People" Ontology from Search Engine Knowledge and DBpedia

One issue in this part of work was how to determine a selection of famous people, as the ideas of who is famous might. There not be representative was not even a preconception of how many people should be considered famous.

Turning these questions to Google itself, about 3200 people were found that are currently famous in the USA by selectively mining Google's suggested completions. This idea relies on utilizing the search engines' public interface, since the public do not have direct access to their query logs.

To collect a wider range of information about these mined famous people, it is necessary to switch to using the already-structured knowledge of DBpedia instead of the mostly text-based Wikipedia. DBpedia is a knowledge base that stores structured data extracted from Wikipedia, and is accessible on the Web [34]. The DBpedia knowledge base currently describes more than 3.5 million entities, including 364,000 persons [16, 34]. DBpedia is considered one of the largest multi-domain ontologies to currently exist. Compared to other hand-crafted ontologies, however, DBpedia is less formally structured. Also, the data quality is lower and there are inconsistencies within DBpedia [63, 64]. Kalender et al. [35] have built an ontology by mapping the Wikipedia instances to WordNet, however, without having any relationship or attribute information. We needed to build an ontology with well-defined classifications and a sufficient amount of useful relationship information to serve the purposes of the OSWS System.

This section presents an approach for extracting useful information about famous people that were determined by mining Google. The information about them is extracted from DBpedia, organizing it as well structured data, and storing it in the ontology of famous people used in the OSWS System. The DBpedia SPARQL interface [65] was used throughout this research, which is publicly available, in addition to the Google Autocomplete SOAP API [66].

3.3.1 Focuses on the New Ontology for Famous People

The OSWS System was designed to provide disambiguated search suggestions; therefore its ontology must contain data useful for such a purpose. The first issue in this research was how to best expand the domain of the OSWS ontology. After exploring the option of

mining Wikipedia data, as in the previous research, instead it was chosen to use the already well-structured data of DBpedia. However, it was found that the DBpedia ontology could not be used "out of the box." There were many types of relationships not relevant to the purposes of the OSWS ontology, redundant relationships (multiple "artist of" relationship types), frequent data errors, as well as inconsistent representations of information (what is an attribute, as opposed to a relationship). To integrate knowledge from DBpedia into the OSWS System subsets of information have to be extracted from their ontology that are compatible for use within the system.

While planning the ontology, three key focus areas of research were identified. The first area was domain coverage; previously the OSWS ontology focused only on American musicians and sports stars (mostly basketball players). This small domain of people is not sufficient for a real-world application, so the goal was defined to expand the ontology and cover as many famous people as possible. However, this required an objective method to determine who should be considered famous in the first place. The second key area was assuring that we store up-to-date, relevant information in the OSWS ontology. It was decided to design a dynamically expanding system that will identify what users of the OSWS System are searching for and, whenever possible, the ontology will be expanded to include those people automatically.

The third focus area was determining DBpedia relationships between classes that are best for use in the OSWS ontology. Due to the nature and purpose of the ontology, it was decided to mainly incorporate relationships from people classes to other relevant target classes, such as movies starred in, songs produced, sports teams played on, etc. This gives a fine granularity for describing people but also provides useful classifications

for other classes they are related to. To develop the OSWS ontology *the Google search engine* was *mined* to identify a small subset of famous people, also using government census data [67] available to the public.

3.3.2 Who is Famous Today?

Here raises the question of who to include in the ontology. Who is famous today? It turned out that Google knows this well, possibly better and certainly more dynamically than other available sources of knowledge. It was hypothesized that famous people are characterized by other people asking questions about them. Thus, it was necessary to find out who those famous people are, even though the public has no access to the Google query logs. However, Google's query logs appear to inform its suggested completions, which helped to develop the following process for mining "famous people" from Google's search suggestions.

Using publicly available US census data [67], the top 1,000 male and female first names (the most common first names in the US) were extracted from the year 2,000 census data. These names were passed to Google, one by one, and the Google responses were recorded by the program. For example, the mining program passed the first name *Robert* to Google and then extracted the last names *Frost, Pattinson, DeNiro, Half* and *Downey Jr.* as people that Google knows are famous right now (Figure 3.3). The returned results were collected and looked for the ones of the form "*n1 n2 n3*," where *n1* is the person's first name, *n2* is the optional middle initial, and *n3* is the last name. The last names were checked against the 5,000 most common last names from the US census

database [67]. With this method, 5,286 potentially famous people were mined from Google.

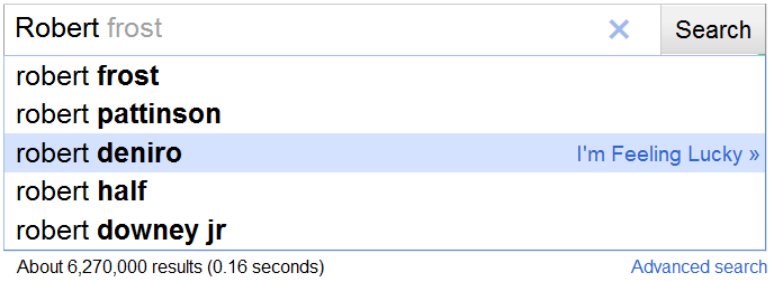

Figure 3.3 Google suggestions for search term "Robert."

It is designated that these people as members of the "A-List," as they are the search suggestions returned by entering only a first name. This process was repeated with "*n1 l1*" style queries, where n*1* is a first name from the census data and *l1* is a letter from the alphabet. This type of query further refines the suggestions by potentially including a specific middle initial or the start of a last name. The results from this set of queries were named the B-List, which comprised of 132,896 candidates. Finally, the program queried the search engine with a series of inputs of the format "*n1 l1 l2*," where *l1* and *l2* are letters in the alphabet, and mined the returned names, storing them as the C-List, composed of nearly a million potentially famous people.

As many properly formatted suggestions were clearly not referring to people (for example, *Sterling Silver*, *Joseph A. Banks*, *John J. College*), it was mandatory to identify which names correlated to an actual person. For this purpose, the program passed the 5,286 names in the A-List to DBpedia to determine for each name whether it refers to one or several people. The program analyzed the type and Wikipedia subject data stored within DBpedia to make this determination. For example, if a given DBpedia page contained the type "ontology:person" or "yago:person," the program considered this a

valid person. Similarly, if the name belonged to the Wikipedia category “Living people” or belonged to Wikipedia categories that end in “Births” or “Deaths,” i.e., “1986_births,” it considered the name to be that of a real person. Using this method, 3,241 famous people were identified among the names in the A-List.

3.3.3 Classification of the Famous People

3.3.3.1 DBpedia Ontology. The OSWS ontology was based on the person hierarchy within the DBpedia ontology. The DBpedia ontology is built from data stored on Wikipedia pages. It forms a shallow subsumption hierarchy [63]. Specifically, the DBpedia designers use the “Infoboxes” which are included on many Wikipedia pages. Infoboxes are tables of attribute-value pairs that are located on the top right-hand side of these Wikipedia pages [63]. These boxes have specific types associated with them, such as “Actor infobox” or “MusicalArtist infobox.” DBpedia’s ontology is built using these infobox types as class names, and a page with a specific type of infobox is assigned that type. For example, *Tom Hanks* has an Actor infobox, and he is an instance of the class “Actor” in the DBpedia ontology. This structure (shown in Figure 3.4) appeared adequate for providing the appropriate granularity for use within the OSWS ontology. Protégé was used as the ontology editing tool.

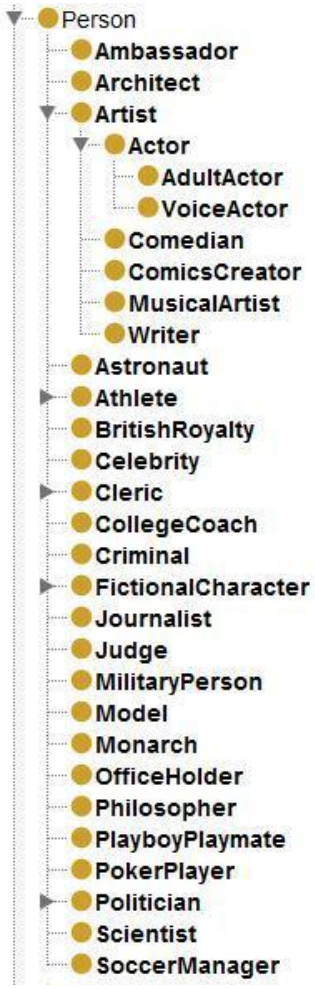

Figure 3.4 Partial "Person" hierarchy in the DBpedia ontology, in Protégé.

The ontology was built by extracting the complete "Person" hierarchy from the DBpedia ontology, and manually adding several other non-person hierarchies. The non-person classes were selected based on the necessity of using them as targets for relationships emanating from the person classes. Once the names of the A-List had been extracted and correlated with real people, instances corresponding to them were inserted into the OSWS ontology.

One major issue when using the DBpedia ontology, however, was that within the A-List hundreds of people were found to exist in Wikipedia who did not have DBpedia

ontology classes associated. A number of ways were developed to augment the DBpedia ontology and to expand the domain coverage. Each correlated name in the A-List was mapped to a class within the hierarchy. While the DBpedia ontology contains over 360,000 categorized people, and continues to expand, there was still a significant number missing, amounting to approximately 520 (17%). Occasionally the DBpedia class was very general and thus uninformative, such as "person." In such cases more specific classes were provided for these concepts.

3.3.3.2 Mapping from YAGO to DBpedia Ontology. YAGO (Yet Another Great Ontology) is an ontology built from Wikipedia leaf categories being mapped to WordNet [68] synsets ("synonym sets") [69]. Because of the way the YAGO ontology was built, instances often belong to many classes. Ontology mapping is becoming increasingly important in the Semantic Web community [70]. The DBpedia ontology provides such mappings to YAGO, and other ontologies. First the DBpedia ontology types and YAGO "rdf:types" were collected for all pages correlated to a name in the A-List.

The YAGO types found in this way were often far too specific for mapping (e.g., "AmericanDanceMusicians"). Types that are too specific defeat the purpose of a classification, as they are unlikely to occur in a Web search and likely to have very few instances. Thus, it was needed to go to their broader types ("Musician" in this case), which provided a more useful class name. With programmed string matching, it was possible to match about 40 YAGO classes to DBpedia classes.

Additionally, the study used the approximately 450 pages which existed in both DBpedia and YAGO to perform a statistical analysis and map YAGO classes to DBpedia

classes. For example, if a certain percentage of pages with the DBpedia ontology type "MusicalArtist" had the YAGO types "Singer" or "Soprano," both of them would be mapped to the DBpedia class "MusicalArtist." Finally, a small number of the YAGO classes were sorted through by hand and mapped to DBpedia classes. In total there were 85 of the most commonly found YAGO classes mapped to DBpedia classes. A sample of YAGO types to DBpedia ontology mappings can be seen in Table 3.1. Most mappings are either identical (Actor to Actor) or consist of more specific YAGO types mapped to less specific DBpedia types. The future work involves augmenting the person hierarchy with the YAGO types which appeared most frequently, in order to provide more specific classifications for people.

Table 3.1 A Sample of YAGO to DBpedia Ontology Mappings

YAGO Class Name	DBpedia Ontology Mapping
Actor	Actor
Anthropologist	Scientist
Biologist	Scientist
Biographer	Writer
Blogger	Writer
Drummer	MusicalArtist
Guitarist	MusicalArtist
Admiral	MilitaryPerson
Marine	MilitaryPerson
Singer	MusicalArtist

For each page with a set of YAGO classes, the more general YAGO classes were determined. Using the mappings, the DBpedia class was found for each YAGO class on the page. Then the occurrences of each DBpedia class were counted and the one with the maximum number of occurrences was selected as the correct mapping class. For example, the YAGO ontology classes for *Kurt Cobain* are "American Diarists," "Grunge Musicians," "Musicians From Washington (U.S.State)," "American Musicians Of Irish Descent," and "People From Olympia, Washington," among others. The more general classes are "Diarist," "Musician," "Musician," "Musician," and "Person," respectively. Our mapping system maps "Diarist" to "Writer," "Musician" to "MusicalArtist," and "Person" to "Person" in the DBpedia ontology. In the above example, *Kurt Cobain* has one mapping to the class "Writer," three to the class "MusicalArtist," and one to "Person." The program chose the most common mapping, in this case "MusicalArtist," and assigned the instance, in this case *Kurt Cobain*, to that class.

Using instances in the A-List that have both YAGO and DBpedia ontology types, this method resulted in the same classification for 268 out of 401 (67%) instances. Additionally, this mapping system determined a more specific class for 47 people (12%) who were classified as "Person" in the DBpedia ontology. Many other mappings were less specific than the DBpedia ontology type given (such as being mapped to "Athlete" instead of "Wrestler"), but still usable.

3.3.3.3 Mapping from Disambiguation Tags to DBpedia Ontology. A large number of pages not categorized by the DBpedia ontology were homonyms of more famous people with the same name, such as Michael Jackson the anthropologist, who is

classified as a "thing" in the DBpedia ontology. As homonyms are a common occurrence in their data, Wikipedia handles homonyms by adding a "disambiguation tag" to the end of a page name. For example, there are a number of *Michael Jordan*'s in Wikipedia. One *Michael Jordan*, the famous basketball player, has a page name of "Michael_Jordan." Other Michael Jordans have page names such as "Michael_Jordan_(footballer)" and "Michael_Jordan (Irish_politician)," a soccer player and a politician respectively.

The information between the parentheses was utilized to construct a set of mappings from disambiguation tags to DBpedia classes. It was possible to map 50 of the most commonly occurring tags. Many disambiguation tags are in the form of [type]_born_[year] (for example "Footballer_born_1984") or [nationality]_[type] (for example "American_singer"). By matching the type to a mapped tag, it was possible to correctly categorize many pages based on their disambiguation tags. Out of 233 people in the A-List who had no DBpedia class and a disambiguation tag, 118 people (51%) were mapped into the Ontology using the 50 most common tags. By adding more mappings it was possible increase this to close to 100%.

3.3.3.4 Mapping from Wikipedia Abstracts to DBpedia Ontology. Whenever a person's type could not be identified using any of the previously discussed methods, the following approach was used. A Wikipedia abstract is the paragraph that appears at the top of a Wikipedia page. Many of the abstracts of person pages start in the form of "someone is/was something," which introduces the occupation of the person. For example, basketball player *Michael Jordan* has his Wikipedia page introduction starting with "*Michael Jeffrey Jordan is a former American professional basketball player*."

After analyzing the abstract of a page and extracting the occupation information, the occupation was checked against the list of class names. Whenever there was a match between the occupation and a class, the person was assigned to that class. In the example above, *Michael Jeffrey Jordan* was determined to be a basketball player by finding *basketball player* in the abstract and matching it to the "BasketballPlayer" class in the ontology. Many abstracts have a list of occupations separated by commas or "and." For example, *Martin Scorsese*'s occupations are "*American film director, screenwriter, producer, actor, and film historian*." This research only considered the first class that the program was able to match.

Using the described method, 248 new instances out of 473 previously unidentified people were added into the ontology. In a random sample of 50 of these instances, their disambiguation tags were manually compared with the mapped DBpedia classes. It was found that 44 (88%) of the instances were matched correctly. All errors in this sample set were due to the way we check occupations against classes. For example, "Martial Artist" was matched to the class "Artist," and "Personality" was matched to the class "Person." Using a more advanced method of string matching or natural language processing would greatly increase not only accuracy, but also coverage of this method. Future work involves implementing these improved methods of string matching in future iterations of the OSWS project.

3.3.3.5 Choosing the Best Classification. To summarize the above four methods for classifying famous people in DBpedia, the classification of a famous person was choose as follows. For each name in the A-List, the program retrieved the DBpedia ontology

class, YAGO classes, disambiguation tag, and abstract, whenever each existed. While mapping, equal weight was given to the DBpedia class, the mappings from YAGO, and the disambiguation tag. In the event that there are multiple possible mappings, the class that is lowest in the hierarchy (the most specific class) was chosen and the instance was assign to that class. In the event that no DBpedia class, YAGO class and disambiguation tag exist for a person, the abstract was used to classify the person.

It was not possible to do an exhaustive audit of DBpedia and YAGO as part of this project. Thus, certain of their problems might have propagated into the ontology. For example, many athletes are associated with a college and the college is given only as a state name instead of the full college name (e.g., "Colorado" instead of "Colorado State University"). This study attempted to handle as many of these special cases as possible, but it is noticed that problems from DBpedia were being propagated into the ontology. Presently this study does not include any automatic quality control mechanisms to deal with this problem. The whole process of building the OSWS System involved manual and semi-automated testing and checks of portions of the ontology. In the future, it is possible to develop automatic quality-control mechanisms for future versions of the system.

3.3.4 Structuring the Relationships and the Attributes

DBpedia contains a rich set of relationships within their ontology, however for search suggestions many of them are not helpful for end-users. Additionally, information is often not organized well enough for use in search suggestions. The study used the

existing relationship data derived from DBpedia for the ontology, but it had to be restructured for use in the OSWS search suggestions.

3.3.4.1 Identifying Possible Relationships and Attributes After assigning each of over 3241 people within the A-List to an appropriate class, the study proceeded to query DBpedia for the types of relationships each instance possessed. It calculated how often each relationship or attribute appeared, relative to its class. This is shown in Formula (3.1), where *Nr* stands for the total number of people in a given class with a particular relationship, while *N* is the total number of people belonging to the class. Relationships and attributes that appear most often were more likely to be useful in the ontology. Thus a threshold *p* was defined as a criterion for selecting useful relationships.

$$\frac{Nr}{N} > p \tag{3.1}$$

In practice, 50% appeared to be a good value for *p*. Common relationships that were not useful for search results were manually excluded, such as "subject" (Wikipedia category), "label," and others. Finally, a manual review of the remaining 213 relationships and attributes was performed to make sure they made sense for their assigned classes.

3.3.4.2 Organizing the Relationships and Attributes One problem with DBpedia's data is that there are often redundant relationships, for example Actors that have the relationships "dbprop:starring" *and* "dbpedia-owl:starring," which were treated as having the same meaning.

A second issue is that some relationships in DBpedia are lacking in granularity or have multiple meanings. For example, the relationship "writer" can mean writer of a book, movie, song, or television show. The context of a relationship often depends on the class of the source. To distinguish between these semi-ambiguous relationships, many DBpedia relationships were split into two or more relationships in the OSWS ontology. This was done by analyzing the parent classes of the targets that a relationship points to. When a large number of targets are of a few different classes (for example, if 30% of the targets for the "writer" relationship are movies and 40% of the targets are television shows) this relationship was selected as a candidate for splitting. In total the research split up the five relationships that had the most variety of targets with close distributions. They are the relationships "starring," "writer," "produced," "genre," and "musical artist of." These five relationships were split into 15 disambiguated relationships. One example is the "starring" relationship, which links a person to a movie or television show. It was replaced with the "stars in television show" and "stars in film" relationships and one or the other was used depending on the type of the target in DBpedia.

For each relationship that was introduced into the ontology, the type of the target was identified in DBpedia in two ways. One way was to retrieve the parent class of the target in the DBpedia ontology, the other was to identify the corresponding Wikipedia subject categories. For example, if DBpedia contains "Forest Gump *starring* Tom Hanks," while adding this relationship to the ontology, the program determines that *Forest Gump* is assigned the DBpedia class "Film." Knowing that *Forest Gump* is a film, the relationship "Tom Hanks *stars in film* Forest Gump" was introduced into the ontology.

The subject and object were switched in many relationships that exist in DBpedia, which corresponds to using the inverse relationship. It is common in DBpedia to see relationships for actors and musicians in the form of [Movie] *starring* [Actor] or [Song] *performed by* [Musician]. For example, the actor *Tom Hanks* is the target of the relationship "Forrest Gump *starring* Tom Hanks." Since the ontology is person-focused, these relationships were reversed to make the person the subject of the relationship. Thus, in the ontology the relationships would be [Actor] *stars in* [Movie] and [Musician] *performs song* [Song]. As for the above example, the relationship "Forrest Gump *starring* Tom Hanks" was changed to "Tom Hanks *stars in* Forrest Gump."

In the above process, the design choice was made to promote certain DBpedia attributes to full relationships, such as the instruments played by a musician and the comedic genres for comedians. By promoting these attributes to relationships to other classes it was possible to more explicitly show a linking of instances, such as two musicians playing the same instrument(s). There are cases where attributes are represented as a comma separated list of terms in DBpedia. For example, if one musician plays piano and keyboard, his or her instrument attributes would be listed as "piano, keyboard." In the described cases the attribute was broken apart at the commas and make each resulting token its own instance.

For non-interpersonal relationships (where the source is a person but the target is not) the targets were organized in a shallow hierarchy based on DBpedia's ontology. All unnecessary classes were removed in the DBpedia ontology (those that are not targets of any relationships) and the ontology was augmented with a number of new classes.

For interpersonal relationships a problem that had to be addressed was how to handle targets that did not exist in the OSWS ontology. Recursively loading all information for each target would rapidly cover a large percentage of DBpedia, filling up the ontology with knowledge irrelevant to the task at hand. This would also run counter to the idea of only storing famous people in the ontology. A person related to a famous person is not automatically famous, although she might be famous in her own right, e.g., as is the case for Bill and Hillary Clinton.

The solution to this problem was the introduction of "stub" instances for such persons. Like other target instances in the ontology, stubs consist of only a name and an assigned class. This prevents the recursion problem while still including those instances within the ontology. If a stub is later determined to represent a famous person, it will be promoted to a full instance and relationship data for it will be loaded into the ontology. Stubs only containing this minimum of information are not returned as search suggestions by the OSWS System.

3.3.5 Building the Ontology of Famous People

The Protégé Java API was utilized to build the ontology programmatically. First the class hierarchy was built and the required relationships were determined. Then, for each name in the A-List, its parent class was identified and the corresponding instance was inserted. For each instance included, the program queried DBpedia for the necessary relationship and attribute information. For each valid relationship, the target was added as a "stub" instance if it did not yet exist. This resulted in an ontology consisting of 3,241 people instances and over 60,000 relationships emanating from them.

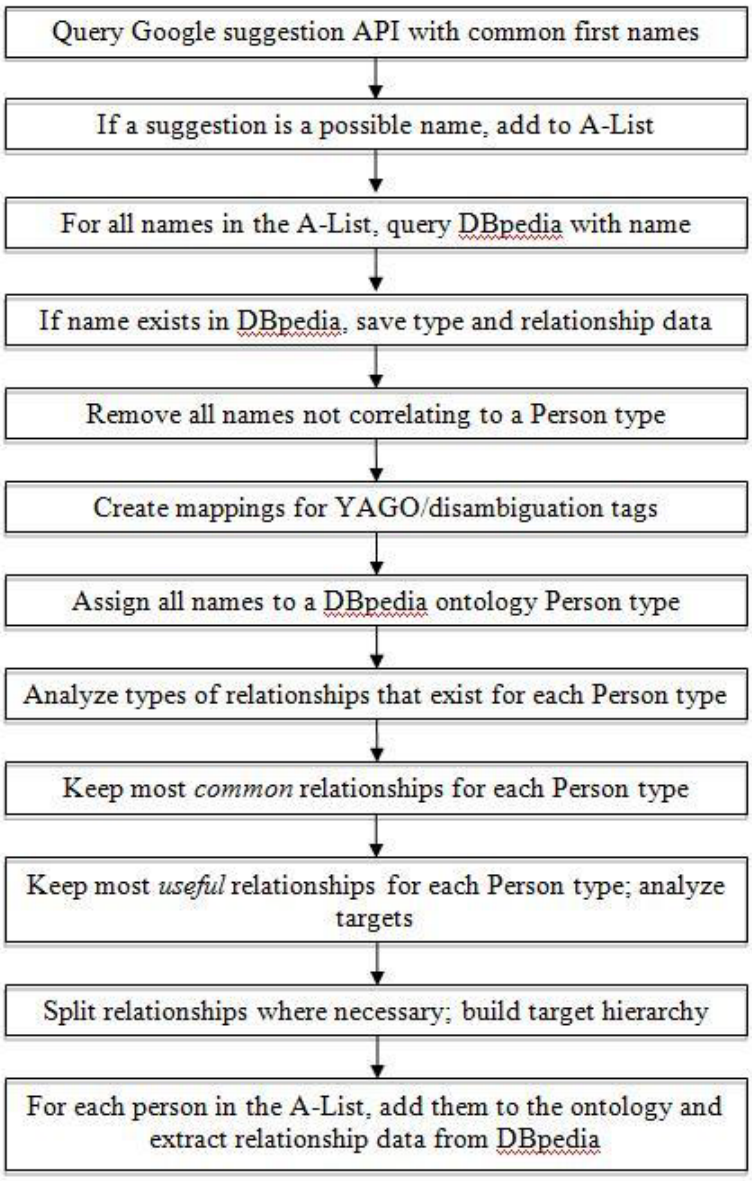

Figure 3.5 Flow of building the famous people ontology.

Finally, this greatly improved ontology was reintegrated into the OSWS front end. The OSWS System includes a number of user friendly features, such as displaying longer versions of relationship and class names (see Section 3.5), class-specific relationships being displayed first, and the ability to filter by class name [6, 51]. The work flow of building the famous people ontology is illustrated in Figure 3.5.

3.4 Dynamically Expanding the Ontology of Famous People

There are two general approaches to ontology development, (with a few exceptions): automatically generated, covering a large domain, or hand crafted, covering a relatively small domain. This is due to the great difficulty of building an ontology by hand. Both of these approaches have disadvantages. Ontologies like DBpedia and YAGO fall into the former. Information stored in these ontologies is generally less well organized and often not reliable when compared to smaller, hand crafted ontologies. On the other hand, handcrafted ontologies are often too small to be practical.

While the A-List ontology covered about 3,200 of the most famous people according to Google (as queried in the Northeast of the USA), it is obvious that users' search interests change on a regular basis. Who is popular and who is not often changes overnight. Keeping an ontology of famous people up-to-date would require significant time and effort if done by hand. To address this difficult issue, the system combines features of automatically generated ontologies and of handcrafted ontologies. For this a system has been developed that automatically keeps the ontology instances up-to-date.

Using the A-List as a "training set," an *expansion system* was developed to dynamically expand the OSWS ontology based on user searches performed with the OSWS interface (Figure 3.6). By plugging the various programs developed for building the A-List ontology into the OSWS front end, a way was devised to expand the ontology with no input from the developers and minimal input of the end-users.

The expansion system works by analyzing user search queries and then including people who are commonly searched for into the ontology. While the expansion system could be used to extract information on the over 360,000 people covered in the DBpedia

ontology, in addition to other people not covered by the DBpedia ontology, this much knowledge would overwhelm both the users and the system itself. *It is in the nature of being famous that relatively few people are famous at the same time.* Furthermore, it is not practical to have potentially hundreds of possible search suggestions for each entered query. Instead, it was decided to focus on providing suggestions only for people who the users consider worthy searching for. If the users of the OSWS System were to query for all of the people within the DBpedia ontology, the coverage of the OSWS System ontology would eventually converge with DBpedia.

When a user enters a name or a certain-sized fragment of it into the OSWS System, it is processed in two parallel threads. In the expansion thread, it passes the query on to Google and retrieves the Google suggested search queries via its SOAP API. The expansion system then checks Google's suggested completions for valid names, using census data for common names, as described in Section 3.3.2. It then queries DBpedia and determines whether any of the possible names correlate to an actual person. If a name correlates to a person then the program determines whether s/he already exists in the ontology. If this is not the case, the program attempts to determine the correct class and creates a new instance of it in the ontology.

If the instance exists in the ontology, but only as a stub, it is promoted to a full instance and then treated in the same way as a new instance. Using the relationships identified previously, DBpedia is queried for applicable relationships and targets. If a given target does not exist in the ontology, a new instance for the target is created before including the relationship. Once the instance has been fully created within the ontology, it is added to the list of valid suggestions and returned to the user, along with any other

previously existing instances that may qualify as search suggestions. The expansion system ensures that OSWS remains up-to-date with search suggestions, and the domain of the ontology expands with no input from the developers.

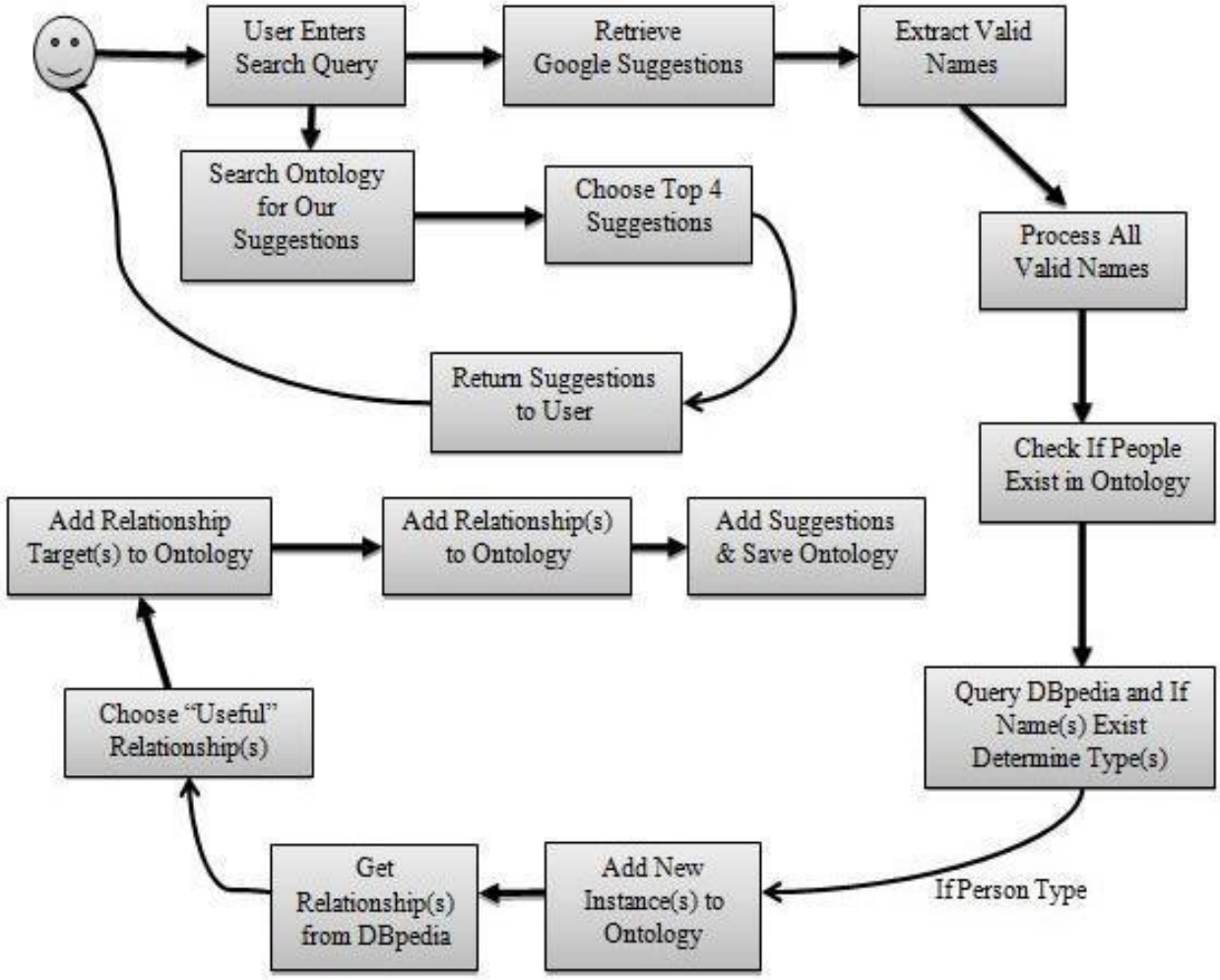

Figure 3.6 Flow of the expansion system.

The process of dynamically expanding the ontology of famous people is diagramed in Figure 3.6. One question that might arise is what value is added by OSWS as opposed to using Google itself. Once information has been extracted from Google into OSWS, the user sees the OSWS search suggestions, as always, distinguished according to homonyms. This service is not provided by Google.

To maintain a reasonably fast response time, the expansion system runs as a background task. There are a number of problems inherent in the described expansion approach. The process of creating new instances dynamically is relatively slow. Also, a single user querying for a person might be looking for his uncle, not a famous person. Thus, it might be recommended only to add a person to the ontology who has been the topic of several queries. Lastly, a person that became famous overnight will not have a DBpedia page yet. Additionally, as there is no interaction with the developers when using the expansion system, the reliability of data obtained dynamically is likely to be lower than the quality of the initially built-in knowledge, which has gone through a partial review by the developers. However, all the dynamically added knowledge is logged by OSWS and may be reviewed by the ontology developers after the fact.

An evaluation experiment of the ontology expansion system was performed. Three independent users were selected to perform in total 100 Web searches for famous people using the OSWS System. Among the 100 input queries, 34 of them already existed in the ontology, thus, their suggested completions were retrieved immediately from the ontology. Another 59 taken from the user queries, did not exist in the ontology and were automatically added.

The remaining seven people had associated Google suggestions but were not found in DBpedia. However, in all seven cases a correct individual existed in DBpedia. In some cases the problem was due to the use of non-ASCII characters. For example, the Spanish name José Luis Rodríguez Zapatero was not matched due to the letter “e” with an accent on top of it. The most common error was not handling “redirects” in DBpedia. For example, *Franklin Roosevelt* was not found in DBpedia, because he was stored as

Franklin D. Roosevelt. However, there is a DBpedia resource named Franklin Roosevelt which redirects to Franklin D. Roosevelt. DBpedia uses this redirect system to handle variations in names. Future work may include using Unicode characters and handling redirecting pages, to better accommodate name variations and various forms of a name.

3.5 Dynamic Ontology-Supported Web Search (D-OSWS)

The D-OSWS (dynamic) search suggestion mechanism remains largely unchanged from the OSWS System. The user is still presented with up to 12 search suggestions comprised of at most four person instances [6, 51]. One change is that instead of using hit counts to determine the display sequence of the homonyms, now the total number of relationships emanating from a given instance was used. The assumption is that the more relationships an instance has, the more popular the person is, since there is more information about this person.

A second change to how search suggestions are generated is based on the *lack* of certain relationship information from an instance. If a relationship is within a class's domain, but a given instance does not have any target for it, the OSWS System provides a search suggestion in the form of

[Instance Name] [Relationship Name].

For example, the system displays the suggestion "Kurt Cobain song," even though there is no song information for Kurt Cobain stored in the ontology (as there is no song information in DBpedia for Kurt Cobain). This search suggestion still might improve the search results, as the relationship name itself is likely to exist in relevant Web pages. This method is only applied to class-specific relationships, to avoid using relationships with

low discriminative power that apply to all people. Another potential issue is using a relationship such as “Died in,” being suggested for a famous person who is still alive.

Search suggestions are now displayed with more user-friendly versions of the relationship names than stored in the ontology. For example, a relationship in the ontology is named “starsInFilm” but the user sees “stars in film.” The same method is also used for class names. This more verbose form of search suggestions distances the end-user from the underlying structure of the ontology and provides information he or she is more likely to understand.

CHAPTER 4

ENHANCING THE FAMOUS PEOPLE ONTOLOGY BY MINING A SOCIAL NETWORK

4.1 Introduction

As described in Chapter 3, an ontology of famous people was built for the OSWS System by retrieving the Google search suggestions and extracting information from DBpedia. Google search suggestions were mined to generate lists of famous people and checked them against DBpedia. If a famous person in the lists exists in DBpedia, his or her DBpedia profile was analyzed and useful information about this person was saved in the OSWS ontology. DBpedia is a huge public resource to serve the purpose of ontology building. However, it is found that many names in the lists do not exist in DBpedia. For example, among the 5,286 names in the A-List, only 3,241 of them were identified in DBpedia.

Besides DBpedia, Facebook [50] was used as a secondary resource to mine information about the famous people who are not included in DBpedia. One of the features of interest in Facebook is the ability to create public pages. Public pages are for organizations, businesses and celebrities to broadcast information about them in an official, public manner [71]. Different from common users, a Facebook public page can be “followed” and “liked” by its fans. Statistics show that “celebrities” is one of the most popular categories among pages with more than one million fans. More and more famous people are using Facebook pages as a marketing tool. They fill out their profile and regularly post new updates on their Facebook pages.

The focus of this part of study was on mining the Facebook celebrity pages in order to increase the size of the OSWS ontology.

4.2 Extending the Famous People Ontology using Facebook

Every page in Facebook has a unique ID. For example, the official page for Barack Obama has the ID 19292868552. One can access the attributes of a page by invoking the Facebook Graph API [72] with the page ID. Thus, as long as the ID of a Facebook page is known, all public information about this page can be retrieved, including attributes like "id," "name," "picture," "website," "birthday," "description," etc. Among all the attributes of a Facebook person page, the most important ones to serve the OSWS ontology are "category" and "likes."

Each Facebook page is associated with exactly one category. All Facebook person pages are under the broad category "person." When creating a Facebook person page, there are 24 subcategories that can be selected. These include categories such as "musician/band," "politician," "athlete," "public figure" etc. For example, Barack Obama's Facebook page has the category "politician." The category information is very useful to classify famous people and to store this information in the ontology. Figure 4.1 shows some of the Facebook person categories.

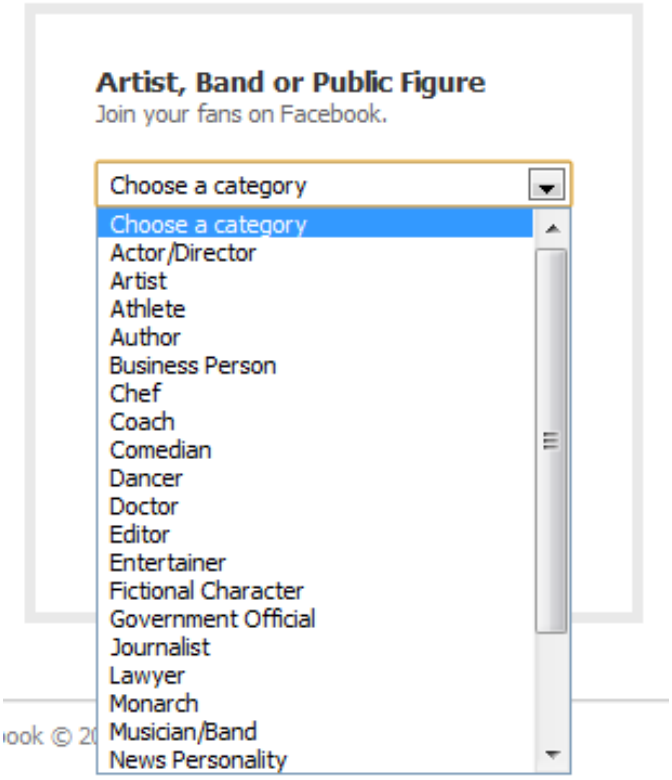

Figure 4.1 Partial of the Facebook person categories.

"Likes" is another useful attribute, which returns the number of users who like this page. In other words, the "likes" attribute represents the number of fans and the popularity of one celebrity. For example, Michael Jackson is the most popular entity on Facebook with more than 14 million fans [73]. Numerous applications have been built to tracking the numbers of "likes" of Facebook pates in order to analyze market trends [74].

4.2.1 Classification of Famous People

It is possible to search over all public objects in the social graph of Facebook. For example, the following search returns the top 50 results of the Facebook pages having "Michael Jackson" in their name:

https://graph.facebook.com/search?q=michael%20jackson&type=page&limit=50.

The attribute-value pair “type=page” expresses the fact that it is intended to search only the public pages. Each returned result contains the name, ID, and category of the page (see Figure 4.2). The first result is in most cases the official Facebook page of the queried famous person.

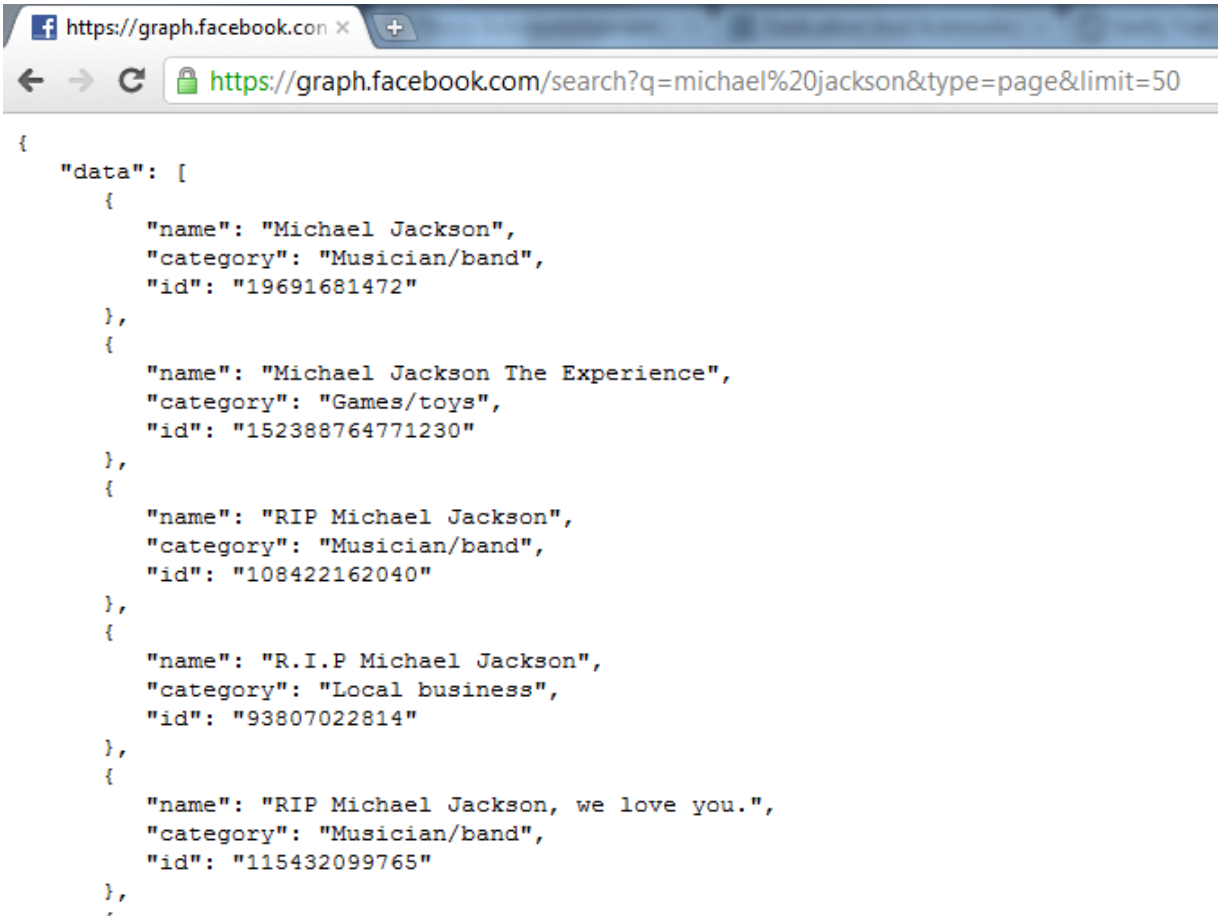

Figure 4.2 Top five results of Facebook pages for query “Michael Jackson.”

Studies began with the 2,564 names in the A-List that could not be found in DBpedia, which was named the “reduced-A-List.” Considering the possibility that the first returned result may not be the official Facebook page when searching for a name, the top 10 returned pages of each search were retrieved. Facebook Graph API calls were made to extract detailed information about those pages. The one with the maximum

number of “likes” was chosen as the selected page, as it was assumed that most of the Facebook users tend to go to the official Facebook pages for updates. Thus, the higher the number of “likes” of a page is, the more authoritative it may be. For example, there are two Facebook pages titled “John Lennon.” One page has 4,795,634 “likes,” while the other has only 1,189 fans. Obviously, the first page is more authoritative and more likely to be the official Facebook page of John Lennon.

Passing the 2,564 names in the “reduced-A-List” to Facebook, 1,894 of them were found to have a Facebook page. By analyzing the categories of these pages, 954 among them were found to be classified as persons. Looking at the results, it is shown that some pages were categorized as person, however, had very few fans. It was necessary to define a threshold of the number of fans to determine which people are important enough to be considered famous, or which page(s) of a celebrity is (are) popular enough so that this person is stored in the ontology.

On average, a Facebook Page has 4,596 fans [73]. Records show that 4% of pages have more than 10,000 fans (for example, as of writing, 32,253 people have liked Cristina Aguilera’s Facebook page), 0.76% of pages have more than 100,000 fans (for example, singer Paul Simon has more than 180,000 fans on his Facebook page), and 0.05% of pages have more than a million fans (for example, Michael Jackson’s Facebook page) [73]. The median page has 218 fans, meaning that 50% of the pages have fewer than 218 fans (for example, as of writing, New York’s local singer Kayla Bliss has 181 fans on her Facebook page.) [73]. In this study, 300 as a minimum number of “likes” was determined for a Facebook page to be selected for analyzing and storing its namesake in the OSWS

ontology. Among the classified people in the "reduced-A-List," 588 of them have over 300 "likes" and were kept for further data extraction.

4.2.2 Extracting Attributes from Facebook Pages

There are some common attributes of Facebook pages, such as "name," "id," "picture," "likes," etc. Also, pages may contain other or additional category-specific attributes [75]. For example, pages of the "musician/band" category may have additional attributes like "album," "genre," "record_label," etc.

There are more than 30 different kinds of attributes among the 588 selected people in the "reduced-A-List." However, many of the attributes are of no use to serve the purpose of providing search suggestions. Examples include the attribute "id," the page's Facebook ID, the attribute "username," the page's Facebook username, the attribute "link," the Facebook link to the page, the attribute "picture," the link to the profile picture of the page, and many other attributes. Those fields are generally Facebook-centric and do not help in suggested completions. Note that the attribute "picture" may be useful in providing detailed information and disambiguating homonymous people. However, the current research focuses only on text-based suggested completions.

Moreover, many attributes are inconsistent and contain "bad" data. For example, attributes such as "description," "bio" and "personal_info" are supposed to hold valuable information about famous people. Since Facebook pages can be created by any Facebook user, however, values to those attributes are in many cases written in casual English,

without a fixed format. Cleaning those attributes will involve a great deal of work with Natural Language Processing tools, which is outside of the scope of this research.

Thus, considering the usefulness of the attributes and after manually checking the quality and trustworthiness of the returned values to the attributes, the following person attributes were chosen to keep in the OSWS ontology. They include attribute "name," "category," "likes," "birthday," "location," "current_location," "hometown," "affiliation" for the category athlete, and "genre" and "record_label" for the category musician/band. Most of the selected attributes need data cleaning and format fixing to fit the purpose of increasing the size of the OSWS ontology. The remaining of this chapter explains the process of cleaning and standardizing the values of those attributes.

In the experiment with the selected 588 people in the "reduced-A-List," 234 people have slightly different names in their Facebook pages than in the list, but the search returns them anyway. This problem mainly results from three sources, the use of Unicode in names and the use of middle initials (recall that the name lists were built with only first names and last names). Besides, some pages are titled with long names containing the person's nickname (for example, "X-Man Toney Freeman") or additional information about the person (for example, "Lynne Curtin of Real Housewife of Orange County").

To solve these problems, firstly the normalization was applied to all the names to transform Unicode characters into equivalent text in English letters. For example, "Penélope Cruz" was converted into "Penelope Cruz." To remove the nicknames and irrelevant information other than names, the names were trimmed so as to keep only the first name, last name and text in between. Exceptions were made when the last names are

followed by “jr,” (stands for junior) “sr” (stands for senior) and numbers such as “II” and “III,” for example, “Marion Barber III.” If the last name is followed immediately by a dash (“-”), both the dash and the word after it were kept, for example, “Kimberly Williams-Paisley.” The text between the first name and the last name (if there is any) is generally filled with the person’s middle name or middle initial. However, in some cases, the page creator added the person’s nickname between the first name and the last name. Examples include DeAndre “Touchdown” Brown, Nicole "Coco" Austin and Hector “Macho” Camacho. It is noticed that the nicknames always appear inside parentheses. Thus, all text between the first and last name and surrounded by parentheses was removed. If none of the above solutions solves the mismatch problem, which occurred only in rare situation, the original names from the “reduced-A-List” were kept.

The values of the attribute “birthday” are standardized in the format “MM/DD/YYYY”, in which MM stands for the month with two digits, DD represents the day with two digits, and YYYY is the year with four digits. In order to present better suggestions, all birthday values were converted to be in the format “Month DD YYYY.” “Month” stands for the full name of the month (for example, January, February, etc.), while DD and YYYY are the same as in the original date.

The attributes “location” and “current_location” convey the same information: the place where the famous person stays at. The value of the attribute “location” is stored in an organized data set with various fields, such as “zip,” “street,” “city,” “state” and “country.” To be consistent with the previous OSWS ontology (Chapter 3), only the city, state (if within USA) and country were saved. The value of the attribute “current_location,” on the other hand, does not have a standard style. A person living in

Los Angeles, California may have the current_location "Los Angeles, CA" or "Los Angeles, California, USA." This does not cause a problem when serving as suggested completion. However, a few cases appear to show irrelevant or even "wrong" information. Common examples include data like "all over," "in the world" and "home." There are two ways to solve this problem. One is to pass the returned location to a search engine, such as Google. If one of the top (five) results shows a link from an official map site, such as map.google.com, then the returned value is a valid location. This method works fine for the "reduced-A-List," but would cause delays when dealing with the much larger B-List (recall the B-List in Section 3.3.2). In this study, yet another solution was used, which is to filter the returned location with a list of stop words. Words such as "all over" and "home," typically do not appear in a valid location. Thus, a manual check of the locations of the "reduced-A-List" was performed. Many elements of the reduced-A-list did not contain any location. From the remaining members the stop words were extracted and included in the stop word list. The values of the attribute "hometown" were cleaned using the same method.

The attribute "category" is the most useful attribute for suggested completions in order to disambiguate homonyms, yet cleaning it causes the most complications. As was mentioned at the beginning of Section 4.2, there are 24 person categories in Facebook. Most of the categories can be mapped directly to the previous OSWS ontology in Chapter 3. "Athlete" is one of the biggest categories in Facebook. In the DBpedia ontology (Chapter 3), the class "athlete" is classified into 22 subcategories of athletes in different sports, such as "BaseballPlayer," "BasketballPlayer," "Boxer," etc. To serve the purpose of providing suggestions for homonyms, it would help greatly if the Facebook "athlete"

category is subdivided to provide more specific information. Thus, more detailed information was extracted from the attributes “bio,” “description,” “personal_info” and “affiliation,” if available. By analyzing the additional attributes and checking for matches with the subcategories of athlete, it was possible to specify 51 people playing specific sports among the 106 athletes found.

Facebook classifies actors and directors into one category: “actor/director.” To determine if a famous person in this category is an actor/actress or a director, it was again necessary to rely on the attribute “bio,” “description” and “personal_info.” If the word “actor” or “actress” was found in those attributes, the person was classified as actor. If “director” was found among the descriptions, the person was specified as a director. If the person was mentioned as both an actor and a director, the occupation that first appears in the description was used. In the last case, if none of the keywords was found, the person was then classified as an actor since there are more actors than directors.

A similar case applies to the category “musician/band.” Since the ontology is about famous people, focus is only on single musicians rather than bands. Once again, there are considerably fewer bands than musicians in the results, because the queries passed to Facebook were combinations of first names and last names. If any singular pronoun was found, such as “he,” “she,” “his,” “her,” “I” and “my,” it is highly likely that the queried name refers to a single musician. If none of the singular pronouns appeared, but the word “band” was found in the description, it is then determined that the information is about a band. Thus, it was removed from the results.

Another problematic category is “public figure.” Any famous person that cannot be categorized as one of the other 23 person categories could be included in “public

figure." Among the 588 selected people in the "reduced-A-List," 164 people are classified as "public figure." Category is considered the most important search suggestion in the OSWS System, because it disambiguates homonyms by their occupations (recall the example of the three famous Michael Jacksons in Section 2.2.2). However, having "public figure" in the suggested continuation does not help much with the disambiguation. Thus, it was necessary to extract "what the famous person does" and "who the famous person is" from the other descriptive attributes, such as "bio," "personal_info," "affiliation" and "description."

During the study, it is found that some Facebook descriptions were copied from various famous persons' Wikipedia abstracts. Thus, using the same method as in Section 3.3.3.4, the person's occupation was extracted by analyzing the Wikipedia abstract. Non-Wikipedia descriptions were checked against a list of occupations generated in this research. This list of occupations contains the other 23 person categories in Facebook, class and subclass names in the DBpedia ontology, and their synonyms in WordNet [68]. The synonyms were retrieved using the Synonym API created by Abbreviations.com [76]. The Synonym API is based on REST (Representational State Transfer) calls which return well-formatted XML results, providing synonyms based on the WordNet database. If more than one matching occupation was found, the first one that appears in the description was chosen. By using this method, it was possible to assign detailed occupation information to 52 famous people with the "public figure" category.

Musician and athlete are two of the largest categories in both the DBpedia ontology and among the Facebook person categories. For the category-specific attributes, the attributes "affiliation" for athlete and "genre" and "record_label" for musician were

kept. The attribute “affiliation” of athlete generally carries information of the team the athlete is playing for. A list of stop words was built through manual checking, to remove noise from the data. The attributes “record_label” and “genre” were processed with the same filtering method. “Record_label” provides information about the company that manages the musician. “Genre” describes the type of music the musician plays. If more than one genre of music occurred, they were separated by various delimiters.

4.2.3 Mapping Facebook Attributes to the DBpedia Ontology

Section 4.2.2 described how the relevant Facebook attributes were selected and cleaned. To integrate the Facebook data into the existing OSWS ontology, it is necessary to find a way to effectively map the Facebook attributes to the DBpedia ontology. The study started with mapping the Facebook person categories to the hierarchy of the DBpedia ontology. It is found that more than half of the Facebook person categories could be mapped directly or indirectly by string and synonym matching to classes in the DBpedia ontology. Table 4.1 shows the mapping from Facebook categories to DBpedia ontology classes.

As can be seen in Table 4.1, not all Facebook categories have exact matches in the DBpedia hierarchy. A number of categories, such as “business person,” and “chef,” “teacher”, are missing from the mapping. In order to fully integrate the Facebook categories, the hierarchy of the DBpedia ontology was manually expanded by adding the remaining Facebook categories. “Director,” “dancer” and “entertainer” were placed as subclasses of “artist,” while “business person,” “chef,” “doctor,” “lawyer,” “news personality,” “producer” and “teacher” were inserted as distinct classes under “person.”

In total, ten new categories were added to the DBpedia ontology. A part of the newly expanded person hierarchy in Protégé is shown in Figure 4.3.

Table 4.1 Facebook Person Categories to DBpedia Ontology Mapping

Facebook Person Category	DBpedia Ontology Mapping
Actor	Actor
Artist	Artist
Athlete	Athlete
Author	Writer
Coach	CollegeCoach
Comedian	Comedian
Editor	Writer
Fictional Character	FictinalChracter
Government Official	Governor
Journalist	Journalist
Monarch	Monarch
Musician	MusicalArtist
Politician	Politician
Public Figure	Celebrity
Writer	Writer

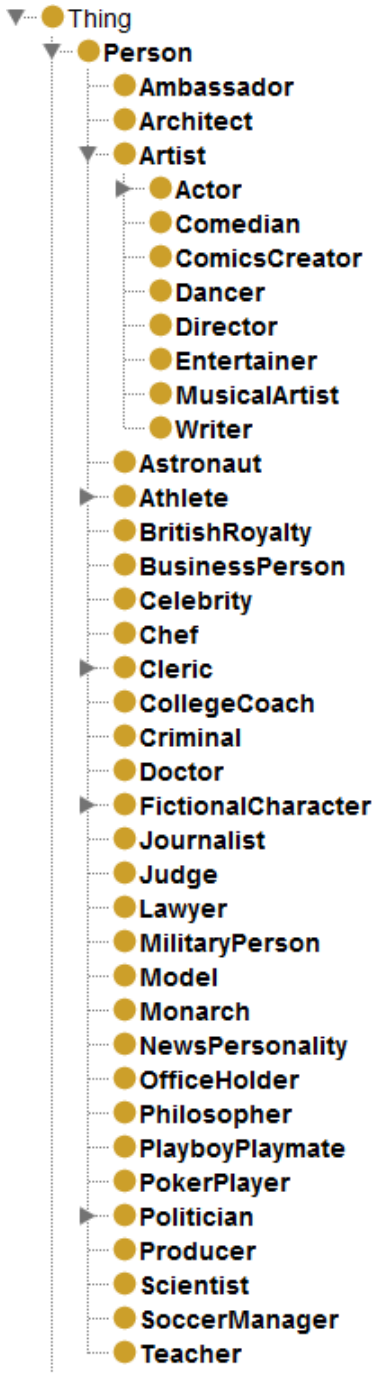

Figure 4.3 Partial view of the expanded "Person" hierarchy in Protégé.

The other Facebook attributes were also mapped to the OSWS ontology, as seen in Table 4.2. The third column in the table shows the type of property (data type or object) when saved in the OSWS ontology. The attribute "name" and "birthday" were stored as

data properties. The remaining attributes were mapped to the ontology as objects, thus, it is necessary make sure that no repetition of objects exists in the ontology. An object property was only added if it did not exist in the previous OSWS ontology.

Table 4.2 Facebook Attributes to "Famous People" Ontology Mapping

Facebook Attribute	"Famous People" Ontology Mapping	Type of Property
name	name	datatype
birthday	dateofBirth	datatype
likes	facebookLikes	datatype
location	currentPlace	object
current_location	currentPlace	object
hometown	placeofBirth	object
affiliation	playsForTeam	object
genre	musicalGenre	object
record_label	recordLabel	object

The processing of the previous version of the "famous people" ontology used the number of relationships and attributes to determine the popularity of a famous person. The Facebook number of "likes" provides the same measurement, but at a different scale, meaning the numbers cannot be combined. Thus, a separate new data type property "facebookLikes" was created in the ontology to store this information. Similarly, another new object property "currentPlace" was added to store the data of the Facebook attributes

"location" and "current_location." The Facebook attribute "affiliation" for athletes was mapped to several properties in the "famous people" ontology, including "playsForBasketballTeam," "playsForBaseballTeam," "playsForFootballTeam" and "playsForSoccerTeam," depending on the type of sports the athlete plays. If the kind of sports the athlete plays was not specified, a new property "playforTeam" was created to store the "affiliation" information.

In total, 584 additional famous people in the A-List were added to the OSWS Ontology by mining Facebook, including 263 artists, 109 athletes, 109 celebrities and many other famous people from the remaining categories. The distribution of the newly added people is shown in a pie graph in Figure 4.4. Each colored area is marked with the name of the category and the number of people that were found in this category.

This part of work was developed using the Facebook Graph API in Java. Unfortunately, Facebook allows only a limited number of Graph API calls per minute. Thus, a timer was set in the programs to send out one API call every two seconds.

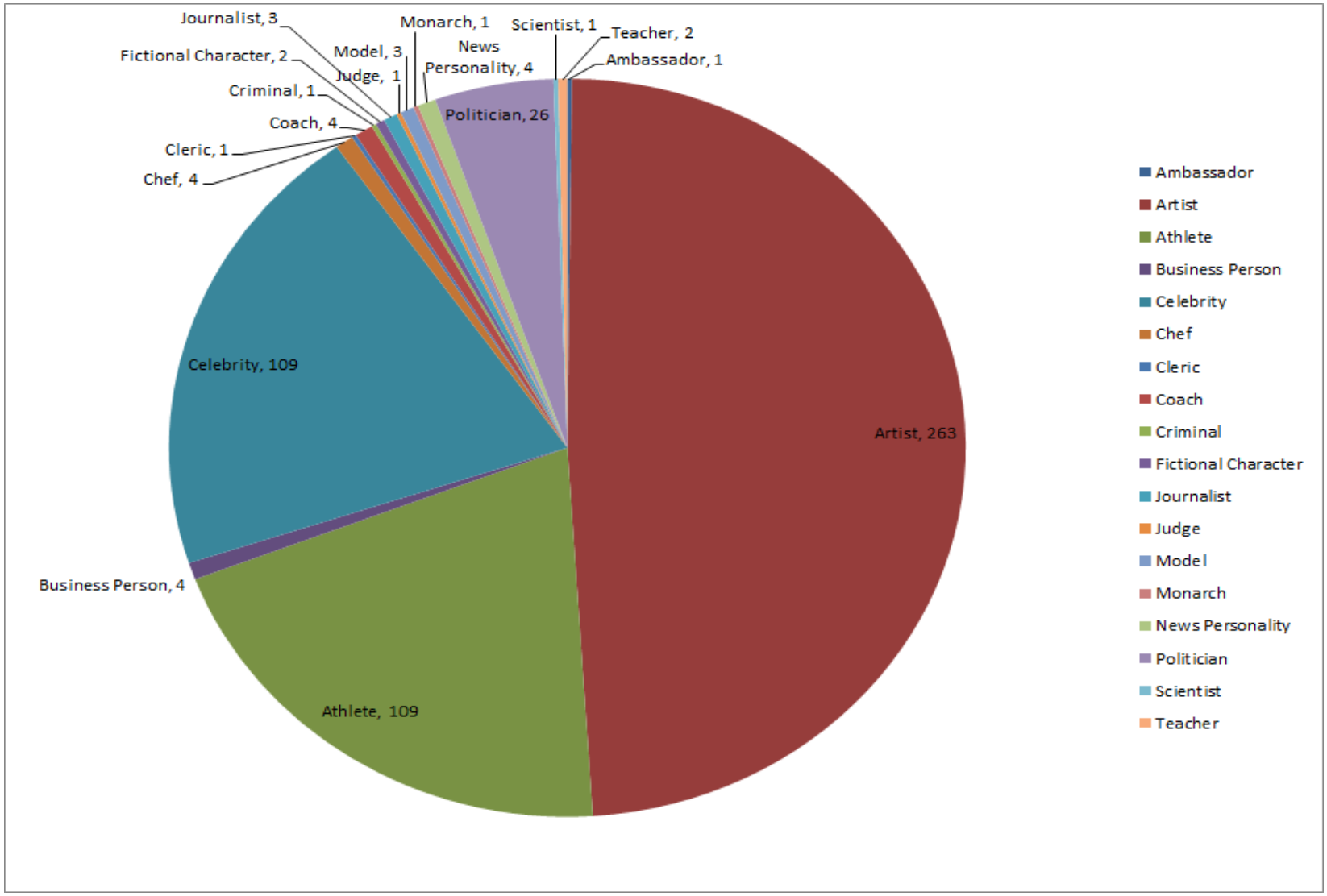

Figure 4.4 Distribution of the newly added famous people to the OSWS Ontology.

CHAPTER 5

PREDICTING WEB SEARCH HIT COUNTS

5.1 Introduction

This chapter is done based on work published in [5] and [77]. As mentioned in Section 1.2.3, it was suggested to add additional terms automatically to the user search terms to get reasonably sized result sets. The additional search terms are retrieved from the same ontology used for disambiguation of homonyms (see Chapter 2). To avoid unacceptable running times by trying too many combinations with additional search terms, the system predicts the number of results returned and only runs searches with expected reasonable result numbers.

Search engines do not guarantee exact numbers of page hits; the total count of results is a rounded estimate of the actual number of results for the search request [78]. Google estimates are sometimes rounded to multiples of 10, 100, or even 1000 [79]. They provide exact numbers of page hits only in cases where these numbers are relatively small [78]. This rounding is probably done because computing exact predictions is expensive if the index is distributed and continually changing, as is the case for large data sets [78]. In Uyar's investigation, compared to other search engines, Google provides the most accurate estimates for document counting. It provides less than a 10% error in 78% of queries for a single-term query experiment [24]. Yahoo provides very accurate estimates for almost half of the queries, but it gives very inaccurate results for the rest [24]. Bing (previously called Live Search) provides a smaller number of accurate

estimates than Google and Yahoo, but the degree of estimate inaccuracy is smaller compared to Yahoo [24]. Bing (Live Search) gives reasonably accurate estimates of the total number of matching URLs with high initial page count estimates (over 8000) [11].

Another potential problem is instability of hit count estimates. The indexes themselves are too big to be stored on one machine and are spread across multiple ones [80]. For availability and efficiency reasons, multiple copies of the same part of the index are kept, which are not always synchronized, since the different copies are updated at different times [78]. As a result, it is possible to connect to different physical machines and get different results for the same query [78]. This is known as search engine "dancing" [78,81]. Uyar has studied the consistency of search engine estimates by observing the fluctuations in estimates over time [24]. Among the three search engines, Google results have the least amount of fluctuations [24].

It would be difficult to have a human experimenter send thousands of interactive queries to Google (Google.com), thus we are using the Google Ajax Search API[2] instead. This API is the substitute for the previous Google API, after Google partially discontinued supporting it [84]. The estimated hit count of a Google query can be retrieved using the Google Ajax Search API. We are not aware of a study about the accuracy of the hit count estimates of the Google Ajax Search API, but there are many documents describing investigations of the "old" Google API. The Google API and the standard interface Google.com (the one used by humans) vary in range, structure and availability [85]. Because Google Standard performs searches in a much larger and

[2] Unfortunately, the Google Ajax Search API has been deprecated since November 2010 [82]. It will continue to work as per their deprecation policy, but the number of requests one may make per day will be limited [82]. The Google Custom Search API [83] is the new substituted search API.

different index than the Google API [85], the Google API gives much lower hit counts than interactive queries. Kilgarriff reports that a substantial number of API results were one-eighteenth as large as comparable interface results [86].

Google is not always computing estimates using the actual words specified in the query [78]. Yet another problem of Google is that it often exhibits non-monotonic behavior, i.e., adding more words in the search query may increase the number of hits instead of decreasing it [87]. This study quantifies the monotonicity problems caused by negative and positive search words (see Chapter 5). Yahoo and Bing have similar problems [79].

5.2 Hit Count Prediction Model

The basic idea for predicting the number of search results is based on the assumption that there is a measurable correlation between the frequency of a word in the English language and the number of Web pages returned by the common search engines. Keller & Lapata have demonstrated a high correlation between page hits and corpus bigram frequencies [88]. Many experiments have been performed on obtaining the frequencies for phrases using the search engine's hit counts. Keller & Lapata used the Web to retrieve frequencies for bigrams. Nakov & Hearst [78] studied the use of search engine page hits as a proxy for n-gram frequencies. Yet, it is not known of research predicting the hit count estimates based on word frequencies. Thus, if a mechanism can be found expressing the correlation between hit count estimates and word frequencies, it can be used to predict the hit count estimates of a search engine. A regression–model was used for this purpose.

The approach for deriving the regression-based prediction model is based on a series of experiments that associate commonly used words, passed them as keywords to a search engine. The correlation analysis was performed between the frequency of the search words and the hit count estimates returned from the search engine(s).

For this purpose the 5000 most frequent English words were chose from the Brown Corpus [89].[3] Stop words [90], bigrams and contractions, such as *I'd* were removed, leaving 4632 words. The study sampled the most frequent English words, because the public does not have access to the frequency distribution of the whole BNC. The program queried the BNC to determine the frequencies of the words from the Brown Corpus.[4] Automated *hit count extraction programs* was developed to send the query and extract hit count estimates from search results, using the Google Ajax API, Yahoo! API and Bing API. The decision to use the most common words was made because the observed frequencies decrease dramatically for infrequent words, even when using a large corpus.

5.2.1 Correlations between Term Frequencies and Page Hit Counts

To derive the prediction model for one-word search terms, the hit count estimates were extracted for many one-word search terms of different frequencies. The hit count extraction program was used to send one search term at a time as input to the Google Ajax API, Yahoo! API and Bing API. The returned page hit estimates were recorded. Zipf observed [91, 92, 93] that given some corpus of natural language utterances, the

[3] The Brown Corpus was used to extract the 5000 most common words because BNC does not provide a word ranking list.

[4] The BNC was chosen as the frequency source because after comparing the results using the Brown Corpus frequencies and the BNC frequencies, it was found that the latter gave much better results.

frequency of any word is inversely proportional to its rank in the frequency table. Zipf's law is most easily observed by plotting the data on a log-log graph. Both the word frequencies and the hit count estimates were converted to log scale. Based on this, a trend line with a second degree polynomial equation was calculated. The same was done for Yahoo! Search and Bing. Thus, there are three separate-prediction models for each experiment.

The same analysis was applied to search terms consisting of several words, with or without negative terms. The probability of a combination of terms can be computed, using standard formulas, based on the frequency list of single words. For example if *A* and *B* are both positive words, then the probability of the term *'A B'*, $Pr(A \cap B)$ can be estimated as in Formula (5.1).

$$Pr(A \cap B) = Pr(A) * Pr(B) \tag{5.1}$$

where $Pr(A)$ is the probability of *A* and $Pr(B)$ is the probability of *B*, both of which are estimated by using the frequency list. Importantly, this assumes that *A* and *B* are statistically independent, an assumption that is commonly made for this kind of analysis [94, 95, 96, 97].

For example, Patel and Lin proposed a term extraction algorithm 'assuming words are independent' [94]; Szpankowski has proposed one model for digital data structures in which 'all words are statistically independent' [96]. Srikanth [95] writes about the independence assumption:

The term independence assumption is used to approximate the query-likelihood probability to a product of query-term probabilities based on the unigram language model. While such an independence assumption is far from reality, it does result in good performing information retrieval models. Language models that incorporate term dependence in the form of N-gram with N > 1 have been explored for retrieval [98, 99]. However, limited improvements in retrieval performance by models using these additional features have raised questions about their contribution towards estimating relevance.

If *A* is a positive word while *B* is a negative word, $Pr(A \cap B)$ will be computed by Formula (5.2).

$$Pr(A \cap B) = Pr(A) * (1 - Pr(B)) \quad (5.2)$$

The same method can be extended to searches with combinations of more than two words.

Yossef and Gurevich sampled queries from a query pool according to their volume distribution [33]. This research selected the words using the uniformly distribute sampling. The case of one positive and one negative term is explained as an example. 250 words were sampled from the word list, evenly distributed among the sample data set. Avoiding the use of the same word appearing as both a positive and a negative term and assuming that '*A B*' and '*B A*' return similar results, *250*249/2!=31,125* pairs were generated. After the trimming, *29,587* pairs remained.

The system uses at most five positive search words and five negative words to keep processing times tractable. The experiments in this research indicated that the correlation between word frequencies and hit count estimates is different for different numbers of search words in a query. Therefore, 30 different data sets were generated with search terms of different lengths and computed the correlations for all of them. In other words, a prediction regression model was generated for one positive search word, another one for two positive search words, etc., up to five positive search words. Then each one of these cases was combined with one negative search words, resulting in five more trend lines. This process was repeated for two, three, four and five negative search words. Thus, thirty cases were analyzed; a separate correlation function (trend line) was derived for each case.

Figure 5.1 shows the scatter plot generated based on the term frequencies and the Google hit count estimates for the case of one positive word. The X axis represents the frequencies of the search terms and the Y axis measures the corresponding Google hit count estimates. Both are in logarithmic scale. It was noticed that many outliers appear when the frequencies or the hit counts are relatively low or high. Thus, a trimmed mean method was applied [100]. The terms with the lowest 1 percent and the highest 1 percent frequencies were discarded. The terms with the lowest 1.5 percent and the highest 1.5 percent hit count estimates were as well discarded. As a result of the trimming, 4.94 percent edge data were eliminated before the correlation analysis.

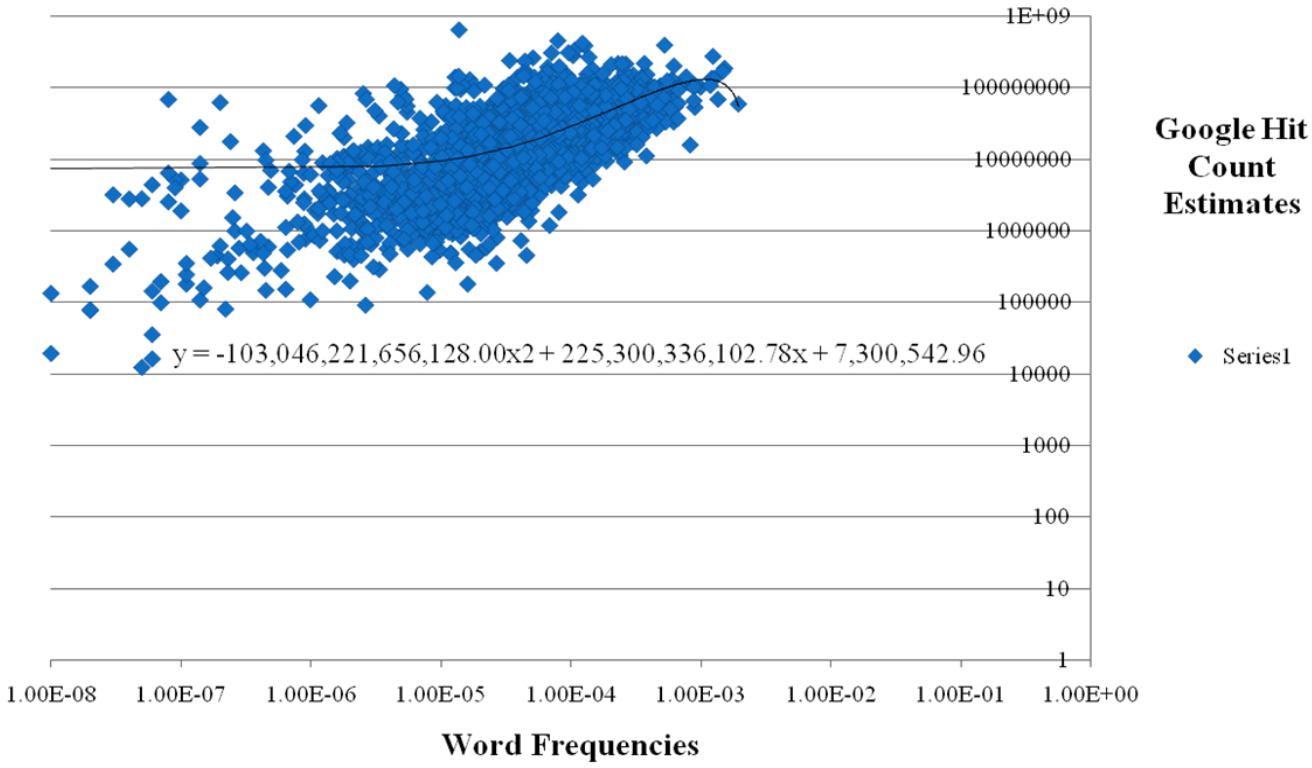

Figure 5.1 Scatter plot of word frequencies and Google hits in log-log scale for case of one positive word.

The program generated 30 second degree polynomial equations based on the logarithmic values of the frequencies and the Google hit counts. For example, for the case with one positive term, the corresponding equation would be as shown in Formula (5.3).

$$H = 0.226 * F^2 + 2.672 * F + 14.415 \quad (5.3)$$

In Formula (5.3), *H* is the logarithmic value of the estimated Google hit count and *F* is the logarithmic value of the combined word frequency. The equation in Figure 5.1 uses the coefficient values before applying the log operator.

All the 4632 words were used for learning the correlations between the term frequencies and the hit count estimates. Constructing all pairs of words, or worse, all n-tuples, from words in this list would put considerable stress on the computational resources and would be impossible for larger values of *n*. Therefore, samples were selected as follows.

For example, the program selected 250 sample words that are evenly distributed, for 2-word samples. This covers two learning conditions (1) two positive words and (2) one positive and one negative word, with the positive word always coming first in the term passed to the search engine. In the same manner, 60 words were used for 3-word learning conditions. Next, 30 words were selected for 4-word cases. Table 5.1 shows the number of words used and the experiment size for each case. Except for the experiment using one word, the size of the rest of the experiments is maintained to be close to 30,000. With the selected sample words, search terms consisting of different combinations of positive and negative words were generated.

Table 5.1 Number of Sample Words Used and Experiment Size for the N-word Cases

No. of Terms (N-term)	No. of Sample Words Used	Experiment Size
1	4632	4632
2	250	29587
3	60	32529
4	30	26051
5	23	31987
6	20	36845
7	18	30252
8	17	23109
9	17	23109
10	18	41596

Table 5.2 shows the values of Spearman's correlation (*C*) between the term frequencies and the hit count estimates returned by the search engines for all the thirty cases. The cases are named in the format of *a*P*b*N, where *a*P represents the number of positive terms and *b*N represents the number of negative ones ($1 \leq a \leq 5$, $0 \leq b \leq 5$). Thus 2P3N stands for the case with two positive and three negative search terms. Here Spearman's correlation was used because the data is sorted in descending order as ranked data. The results of Table 5.2 verify the initial assumption, that is, for most cases there is a positive correlation between the term frequencies and the hit count estimates returned by major search engines. The p values for all experiments were < 0.001.

Table 5.2 Correlation between Word Frequencies and Hit Count Estimates

	Google's Correlation (C)	**Yahoo's Correlation (C)**	**Bing's Correlation (C)**
1P	0.706	0.684	0.572
2P	0.674	0.680	0.577
1P1N	0.699	0.657	0.636
3P	0.737	0.863	0.814
2P1N	0.701	0.725	0.734
1P2N	0.616	0.626	0.385
4P	0.639	0.863	0.783
3P1N	0.615	0.826	0.839
2P2N	0.550	0.554	0.506
1P3N	0.302	0.348	-0.195
5P	0.594	0.654	0.684
4P1N	0.595	0.663	0.801
3P2N	0.253	0.696	0.801
2P3N	0.508	0.678	0.557
1P4N	-0.042	-0.324	-0.299
5P1N	0.553	0.762	0.718
4P2N	0.294	0.740	0.567
3P3N	0.646	0.635	0.375
2P4N	0.357	0.680	0.408
1P5N	0.435	0.580	0.625

Table 5.3 Correlation between Word Frequencies and Hit Count Estimates (Continued)

	Google's Correlation (C)	**Yahoo's Correlation (C)**	**Bing's Correlation (C)**
5P2N	0.732	0.260	0.465
4P3N	0.488	0.446	0.500
3P4N	0.391	0.469	0.505
2P5N	0.777	0.631	0.556
5P3N	0.770	0.787	0.769
4P4N	0.696	0.773	0.709
3P5N	0.233	0.604	0.534
5P4N	0.631	0.804	0.779
4P5N	0.719	0.739	0.643
5P5N	0.768	0.825	0.800
Mean	**0.555**	**0.631**	**0.572**

5.2.2 Evaluating the Prediction Model

In the experiments, the 10-fold cross-validation method [101] was used to evaluate the prediction module. That means, the data were split into ten "folds" of equal size. Then the data was "trained" with nine folds and its success was evaluated with the tenth fold. This process is repeated ten times, such that every fold is used one time for testing. During "training" a regression line is derived. During testing this regression line is used to predict hit count estimates for terms which were not used during training. The predicted hit count estimates are compared with the hit count estimates reported by the search

engines, one at a time. Ideally, the two numbers should be equal. The same experiments were performed for evaluating the three search engines.

To evaluate the effectiveness of these prediction models, we used the following measures. E_i, the percentage of difference between the predicted hit count estimate and the search engine hit count estimate, is calculated by Formula (5.4), where Pr_i stands for the predicted hit count estimate and Hc_i represents the real search engine hit count estimate:

$$E_i = \frac{|\mathrm{Pr}_i - Hc_i|}{Hc_i} \times 100\ (\%) \tag{5.4}$$

E, the average percentage of error on the test set, is the average value of all E_i*'s* from Formula (5.4), where n is the size of the test set:

$$E = \frac{\sum_{i=1}^{n} E_i}{n} \tag{5.5}$$

To analyze the accuracy of the predictions, we also used *coefficient of variation (CV)*, a normalized measure of dispersion, which is calculated by Formula (5.6) where *SD* is the *standard deviation* of the E_i values. The smaller CV is, the better is the prediction.

$$CV = \frac{standard\ deviation}{average} * 100\ (\%) = \frac{SD}{E} * 100\ (\%) \tag{5.6}$$

Table 5.4 Correlation Summary for Thirty Cases

%	G's E	G's SD	G's CV	Y's E	Y's SD	Y's CV	B's E	B's SD	B's CV
1P	75.0	54.3	87.1	88.4	61.2	85.1	113.0	69.6	76.0
2P	199.5	188.9	120.2	130.3	110.3	89.1	134.3	97.6	89.9
1P1N	94.8	70.6	67.3	110.5	84.1	68.7	103.4	59.9	72.1
3P	235.4	254.0	100.1	378.8	584.4	126.7	144.5	162.2	102.6
2P1N	275.8	279.3	93.1	132.0	126.6	79.0	119.9	121.0	81.9
1P2N	76.3	33.4	56.4	100.6	41.1	55.6	103.4	49.7	48.1
4P	156.7	127.7	80.6	145.7	198.4	114.6	141.2	100.7	82.4
3P1N	272.5	185.4	66.2	127.7	156.8	98.9	143.6	178.1	112.0
2P2N	987.5	1163.1	67.8	248.7	221.3	65.0	242.4	224.7	72.0
1P3N	81.1	45.0	57.9	92.9	60.2	49.1	84.6	18.3	26.1
5P	162.6	166.5	105.3	340.9	743.0	89.5	68.2	45.6	72.9
4P1N	146.6	108.5	74.6	140.1	156.2	82.5	63.3	54.7	89.6
3P2N	188.3	76.8	39.9	86.3	58.3	77.6	70.5	61.7	88.2
2P3N	140.0	86.2	51.6	59.2	39.5	69.5	75.2	48.8	62.3
1P4N	39.8	7.4	18.9	47.5	22.6	52.3	42.8	13.9	51.2
5P1N	125.7	98.5	86.4	189.7	167.1	86.8	120.0	86.7	75.4
4P2N	174.4	124.1	76.9	88.6	68.6	83.3	94.9	60.9	74.4
3P3N	274.6	164.0	76.1	95.0	57.7	65.4	139.2	62.7	56.5
2P4N	387.0	177.4	48.8	79.3	42.4	52.6	131.1	64.7	61.1

Table 5.5 Correlation Summary for Thirty Cases (Continued)

%	G's E	G's SD	G's CV	Y's E	Y's SD	Y's CV	B's E	B's SD	B's CV
1P5N	92.0	40.6	27.4	51.9	24.3	38.3	76.0	32.6	57.4
5P2N	86.6	28.8	44.0	203.5	93.5	51.6	82.4	41.5	61.7
4P3N	116.1	51.4	59.8	167.6	100.5	63.5	95.0	48.2	66.1
3P4N	172.0	93.4	63.8	141.6	68.4	58.8	114.3	57.2	60.0
2P5N	174.6	77.3	54.5	93.6	33.3	45.9	70.3	23.4	43.1
5P3N	88.2	56.5	68.4	137.3	131.0	104.1	79.2	60.4	79.7
4P4N	64.0	51.7	83.5	122.4	102.4	85.7	89.0	50.9	65.0
3P5N	57.0	35.6	72.1	140.8	88.6	70.1	97.2	81.0	65.1
5P4N	66.4	46.9	75.4	122.9	114.4	103.8	80.7	59.0	75.9
4P5N	62.9	49.6	79.8	132.8	98.9	81.5	104.6	60.6	60.4
5P5N	56.5	40.8	77.4	115.6	105.5	100.9	80.3	48.8	71.0
Mean	**171.0**	**132.8**	**69.4**	**137.1**	**132.0**	**76.5**	**103.5**	**71.5**	**70.0**

Table 5.3 shows the average results after applying the 10-fold cross-validation method for all thirty cases. The cases are named in the same format as in Table 5.2. The average error (E in equation (6)), standard deviation (SD) and coefficient of variation (CV in equation (7)) of the three search engines are reported in this table. Due to space limitation, we used the abbreviations in the table header. G, in Table 5.3, stands for Google, while Y stands for Yahoo! and B represents Bing. To be consistent with the variables used in the equations, we chose E to represent the average error, SD for the standard deviation and CV for the coefficient of variation.

Note that the coefficient of variation in Table 5.3 is not simply the result of the corresponding standard deviation divided by the average error in the same row. For example, in the case with one positive term only (1P), the coefficient of variation of Google is 87.1 percent, which is not the value computed from the standard deviation (54.3% in this case) divided by the average error (75.0%). Instead, it is the average of the ten different observed coefficients of variation, since the 10-fold cross-validation method was used during the evaluation. Similarly, the standard deviations and the average errors in the table are the average values of ten sets of validation results.

From Table 5.3, one can see that the statistical results (*E*, *SD* and *CV*) measuring the errors of predictions are in most cases relatively small. These results support the assumption that there is a measurable positive correlation between the frequencies of English words and the hit count estimates returned by three major search engines. Among the three search engines in these experiments, Bing behaved better than both Google and Yahoo! Search, producing error statistics around 100 percent or even smaller. While this might appear as a large number, in the context of the goals of this study, 100 percent is still acceptable. Practically speaking, if the system presents a user with at least 10 and at most 100 Web page hits for a search term, a 100 percent error would imply that there might be 200 hits instead, which is still a manageable number compared to the typical results in the thousands to millions.

The hit count estimates of the three search engines were initially fetched in February 2010 [5]. Considering the possibility of the search engines' dancing [81], that is the same search results in different reported hit count numbers, the reliability of the results was tested by comparing the ones retrieved in February 2010 with the August

2010 hit counts returned by the three search engines. For each of the 4632 queries in one positive (1P) case, the two hit counts fetched in February and August were compared by calculating their difference ratio *DR*, as shown in Formula (5.8). If the search engines were stable during the six month interval, the difference ratio should be around 1 for most of the queries.

$$DR = \frac{Hc(August,2010)}{Hc(February,2010)} \qquad (5.8)$$

Figure 5.2 shows the scatter plot of difference ratios for all the queries in the 1P case for Google, Yahoo! and Bing.

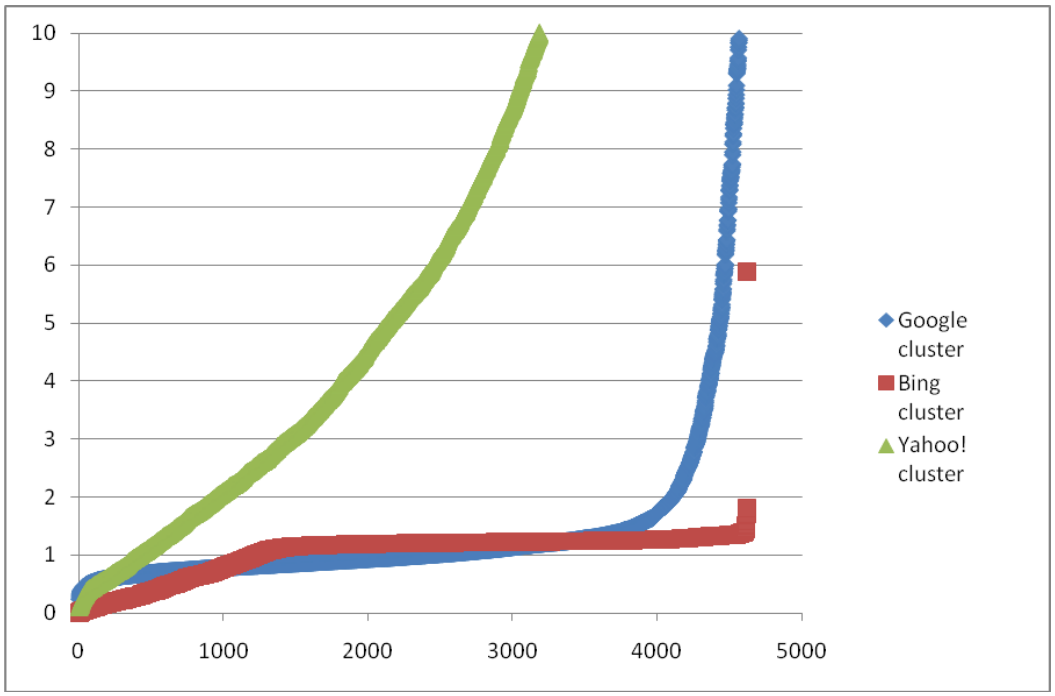

Figure 5.2 Clustering result of hit count transition within a six month period.

One can see that the major portions of Google's and Bing's clusters are around 1. (According to the experiments in this study, the knee in Google's cluster is observed to be normal during Google's stable period.) Thus, it is confirmed that the hit count

estimates by Google and Bing were fetched during the stable period of the search engines. However, Yahoo!'s cluster shows great dancing in this observation. It is presumed that this was the case because Yahoo updated their indexes in March 2010. Thus, Yahoo!'s results in Table 5.3 were replaced by their stable hit counts. To retrieve more reliable hit counts, the program used the hit count estimates appearing on the later search result pages of Yahoo! [81].[5]

The Google Ajax Search API, Yahoo! Boss API and Bing API were used throughout this research. Unfortunately, however, the new Google Custom Search API provides only up to 100 free queries per day [83]. This change of Google has caused great obstacle in continuation of this study.

[5] The offset of 900 was used for Yahoo!'s hit count estimates. It was not possible to retrieve more precise counts from Google because Google provides up to the top 64 results in its API. Bing adjusts its hit counts to very small numbers at different offsets for different queries.

CHAPTER 6

EFFECT OF NEGATIVE AND POSITIVE WORDS IN THE SEARCH

One of the observations made in the study in Section 5.2 was that negative search terms change the hit count estimates in quite an unpredictable way, which has caused practical problems when implementing the prediction model (Section 5.2). This dissertation has investigated this problem with the search engines' behaviors. This chapter is presenting the results from this investigation.

It is assumed to be obvious, that when a negative search word is added to a previous positive search term, then this would exclude some of the results of the positive search term. Thus, the hit count estimates should always decrease when adding a negative search word to a search term. This kind of behavior has been referred to as "monotonicity." However, the experiments in this part of study indicated that all three search engines show non-monotonic behavior for negative terms.

The problem of non-monotonicity is especially vexing because (1) numbers are not just wrong in a quantitative sense, they are qualitatively wrong, increasing instead of decreasing; (2) non-monotonicity contradicts claims made by the major search engine companies, i.e., in Google's, Yahoo's and Bing's documentations [102, 103, 104].

To investigate the scope of this problem, another series of experiments were performed, exclusively focusing on the question whether negative search terms reduce or increase the hit count estimates reported by the search engines. A sample of 12,000 cases was used, which were constructed as follows. Suppose there are random words w_1, w_2,

w_3, w_4, w_5, and w_6 from the sample word list. Then a series of six queries (marked as $query_i$, where $1 \leq i \leq 6$) is constructed as follows:

w_1	$query1$
$w_1 - w_2$	$query_2$
$w_1 - w_2 - w_3$	$query_3$
$w_1 - w_2 - w_3 - w_4$	$query_4$
$w_1 - w_2 - w_3 - w_4 - w_5$	$query_5$
$w_1 - w_2 - w_3 - w_4 - w_5 - w_6$	$query_6$

Each series starts with the positive word w_1 in $query_1$ and is added one more negative term in each query. Moreover, all the possible sequences of combinations of the five negative words were considered. For example, two possible series of queries could be as follows:

w_1	$query1$
$w_1 - w_2$	$query_2$
$w_1 - w_2 - w_3$	$query_3$
$w_1 - w_2 - w_3 - w_4$	$query_4$
$w_1 - w_2 - w_3 - w_4 - w_5$	$query_5$
$w_1 - w_2 - w_3 - w_4 - w_5 - w_6$	$query_6$

and

w_1	$query1$
$w_1 - w_2$	$query_2$
$w_1 - w_2 - w_4$	$query_3$
$w_1 - w_2 - w_4 - w_3$	$query_4$
$w_1 - w_2 - w_4 - w_3 - w_5$	$query_5$
$w_1 - w_2 - w_4 - w_3 - w_5 - w_6$	$query_6$

In total there are $5!=120$ possible series of queries considering all sequences of combination constructed by the five negative words.

According to the major search engines' documentations [102, 103,104], the query "$w_1 - w_2$" returns the Web pages which exclude the term "w_2" from the pages returned by querying "w_1." The same analysis applies when moving from $query_i$ to $query_{i+1}$, $query_{i+2}$, etc. In other words, the hit count estimate of *query* should never be greater than the hit count estimate of $query_j$, when i is greater than j. However, the experimental results in this research show that this is not true for many cases.

These results were encoded as follows. Whenever adding a negative search word decreased the hit count estimate, this was coded with a 1. In other words, whenever the search engines behaved correctly, 'monotonically' and decreased the hit count estimate after adding a negative word, this was coded with 1 (true). When an additional negative search word increased the hit count estimate, this was represented as 0 (false). For example, it is coded by 1 when the hit count of $query_4$ is no greater than the hit counts returned by $query_1$, $query_2$ and $query_3$.

Table 6.1 shows the results of analyzing the Google behavior. The results in *Step 1* (from $query_1$ to $query_2$) reflect the correctness of adding one negative search word to one positive search word. Similarly, results in *Step 2* show the effect of adding two negative search words to one positive word, and so on. In this case the comparison is made between the new case (one positive and two negative words) with both previous cases, i.e., with one positive word only as well as with one positive and one negative word. Similarly, for later steps, the comparison is made with all previous queries.

Table 6.1 Results of Experiment on Effect of Negative Terms in Google Search

	Correct	**Incorrect**
Step 1 ($query_1$ to $query_2$)	1645 (13.7%)	10355 (86.3%)
Step 2 ($query_2$ to $query_3$)	1749 (14.6%)	10251 (85.4%)
Step 3 ($query_3$ to $query_4$)	1903 (15.9%)	10097 (84.1%)
Step 4 ($query_4$ to $query_5$)	2055 (17.1%)	9945 (82.9%)
Step 5 ($query_5$ to $query_6$)	2192 (18.3%)	9808 (81.7%)

As can be seen in Table 6.1, 86.3% of the cases resulted in incorrect behavior when adding one negative word to one positive word. As a result, among the 12,000 sample cases in the experiment, only 1,051 cases (8.8%) can be described to be totally correct. Thus, it can be concluded that Google rarely behaves in the announced way for cases with negative search words.

Figure 6.1 shows the results of this experiment by encoding the strings of 1s and 0s in the following way. Each string of 1s and 0s was interpreted as a binary number.

Then the program computed the corresponding decimal number for each such five-digit string. Next it sorted the resulting decimal numbers in descending order and plot them over the number of cases (12000) of the experiment. If a 0 appears at the left-most position, that means that a single negative term already leads to non-monotonic behavior of the search engine, which is much more serious than if a 0 appears in the right-most position. Therefore, a 0 in the leftmost position should reduce the "correctness" of this case much more than a 0 in the rightmost position.

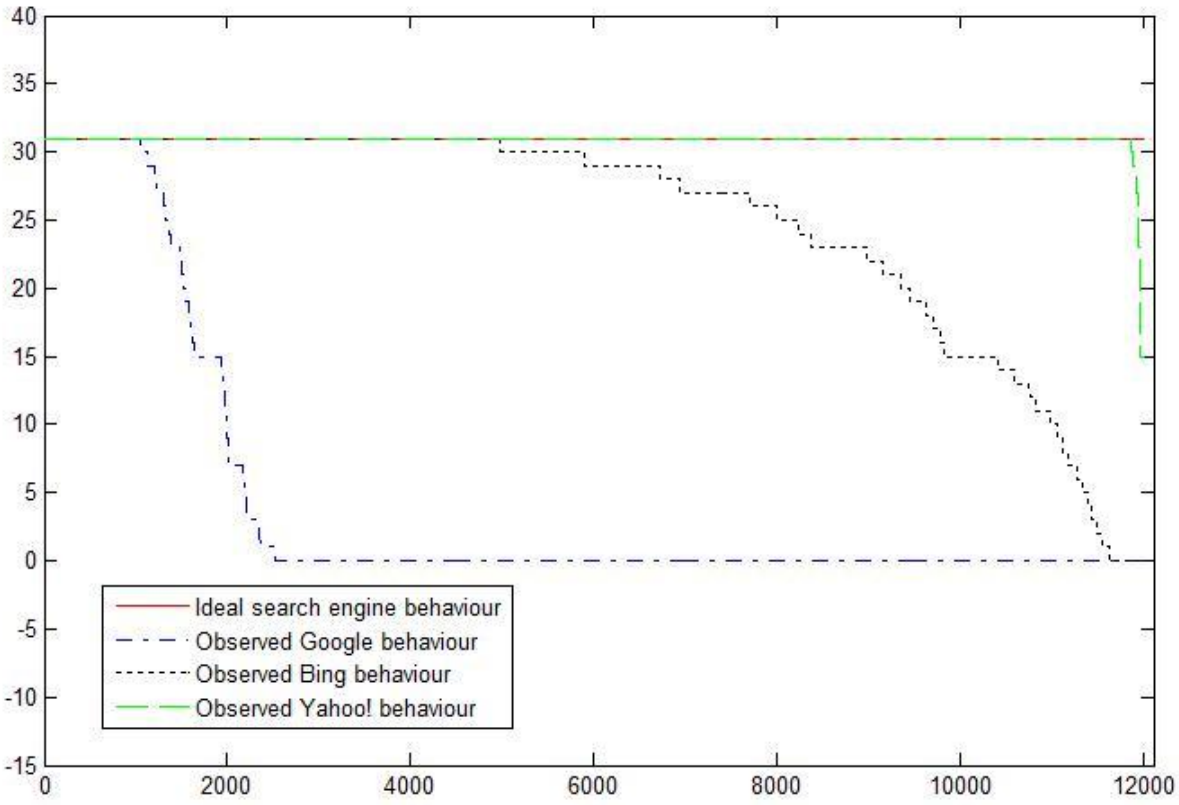

Figure 6.1 Observed search engine behavior vs. ideal search engine behavior (experiment 1: one positive word).

This behavior of giving more weight to the left-most position is exactly the effect of the binary encoding. A 0 in the left-most position will reduce the decimal value by 16,

i.e., to less than half of the possible maximum. In the graph, the horizontal line at level y=31 indicates the ideal case of a search engine behaving correctly for all experiments (11111). Thus, the distance of the jagged line from the horizontal line indicates for every case how far off it is from the correct behavior.

As can be seen in Figure 6.1, Google very rarely behaves in the announced way for cases with negative search words. Yahoo! Search and Bing behave much better than Google with respect to monotonicity. In this experiment, Yahoo! Search behaves the best among the three search engines. Only 1.3% queries showed the monotonicity problem.

An additional experiment was also performed with the three search engines of the effect of negative words on two positive terms. 12000 cases were constructed as follows. Assume that there is another word w_7. Thus, the six queries in one case are:

$w_1\ w_2$	$query_1$
$w_1\ w_2\ -w_3$	$query_2$
$w_1\ w_2\ -w_3\ -w_4$	$query_3$
$w_1\ w_2\ -w_3\ -w_4\ -w_5$	$query_4$
$w_1\ w_2\ -w_3\ -w_4\ -w_5\ -w_6$	$query_5$
$w_1\ w_2\ -w_3\ -w_4\ -w_5\ -w_6\ -w_7$	$query_6$

While non-monotonicity is obviously a serious logical problem with negative search words, it cannot be neglected for positive search words either. Users intuitively expect “AND semantics” when adding more words. Thus the number of search results for a one word query should be reduced when adding a second positive search word.

Unfortunately, it appears that Google is trying to ‘outsmart’ the user by making it hard to reach single digit hit counts even when accumulating rare words from different domains. Therefore, an investigation of this behavior is warranted.

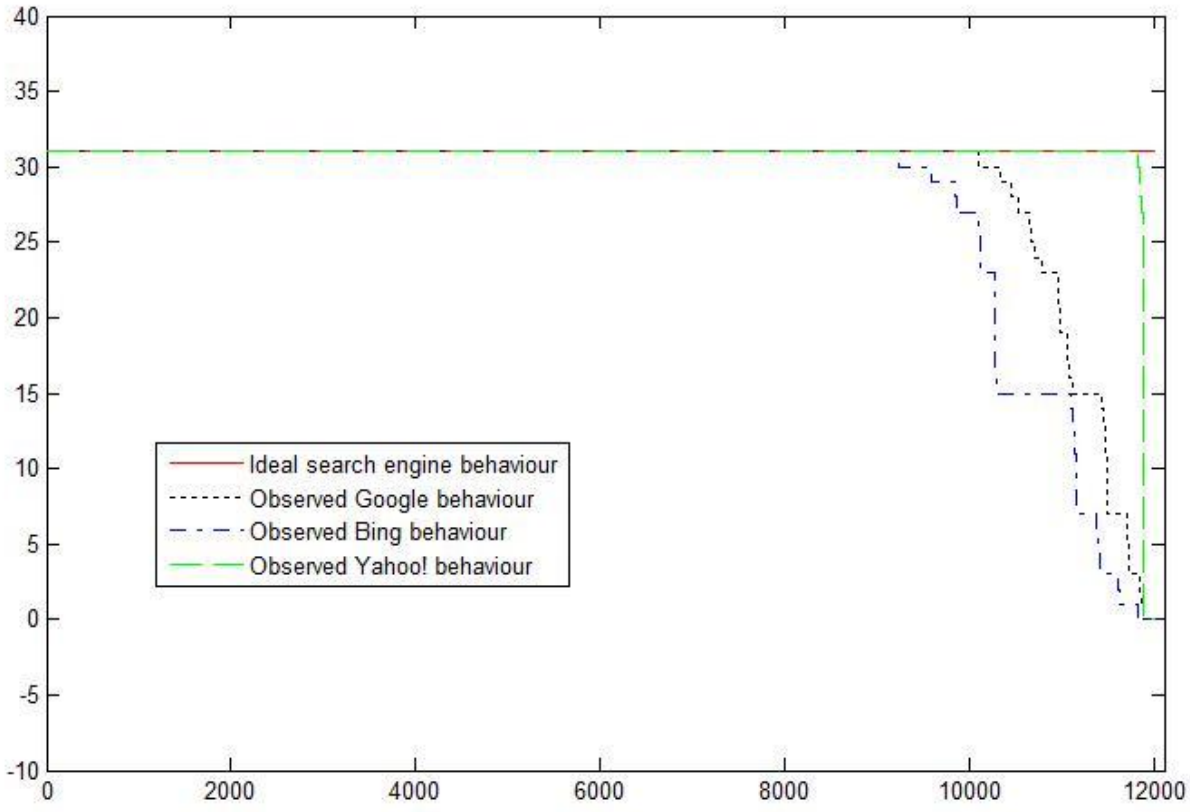

Figure 6.2 Observed search engine behavior vs. ideal search engine behavior (experiment 2: two positive words).

The following experiment studied the monotonicity of appending additional positive search words at the end of a query. That is, taking six random words w_1, w_2, w_3, w_4, w_5, and w_6 from the sample word list, a series of six queries was constructed by appending one additional word to w_1 each time. Similar to the previous experiments, the

six queries are constructed as w_1, $w_1\ w_2$, $w_1\ w_2\ w_3$, $w_1\ w_2\ w_3\ w_4$, $w_1\ w_2\ w_3\ w_4\ w_5$, and $w_1\ w_2\ w_3\ w_4\ w_5\ w_6$.

With the same encoding method used as in Figure 6.1 and Figure 6.2, it is possible to summarize the monotonicity results of adding positive search words for the three search engines. As can be seen in Figure 6.3, Yahoo! and Bing behave much better than Google with respect to monotonicity of positive search words.

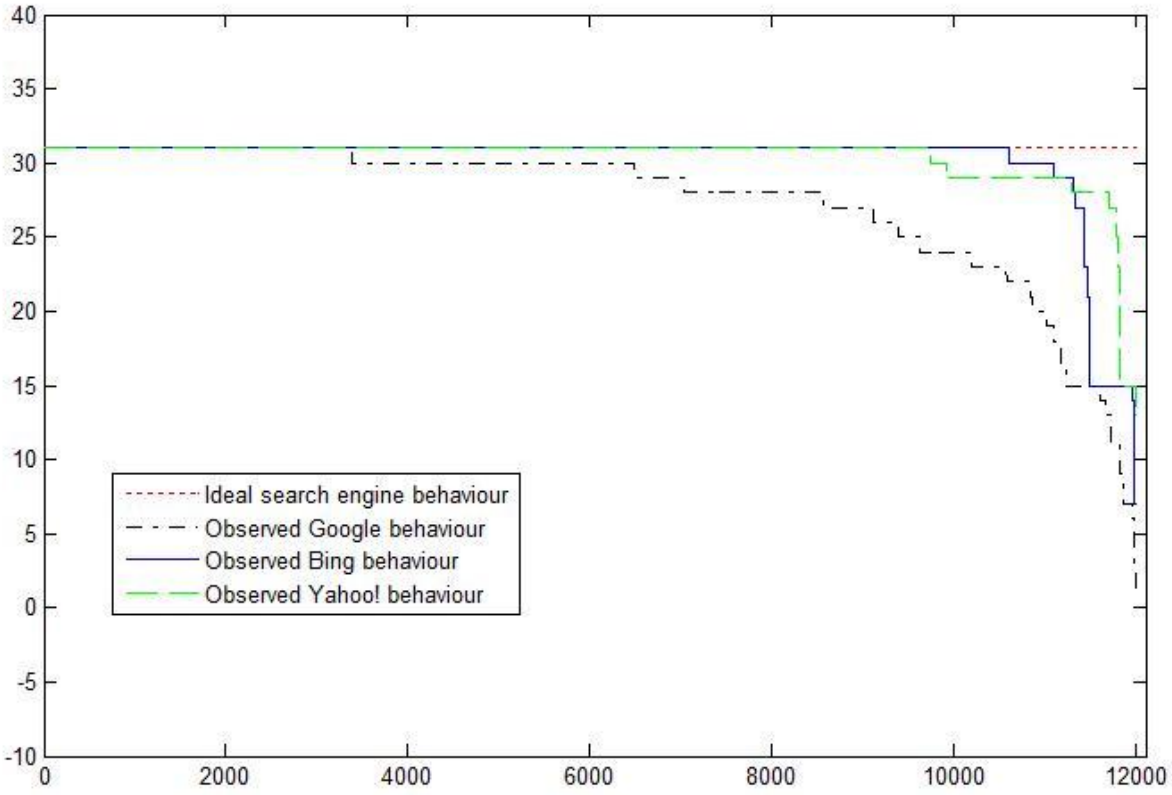

Figure 6.3 Observed search engine behavior vs. ideal search engine behavior (experiment 3: only positive words).

While considering these results as interesting in their own right, this study is part of a larger project of building a browser (plug-in) that controls Web search in a manner that avoids overwhelming users with too many search results but is not so restrictive as to

return no results at all (Chapter 5). To control search results, negative terms are used. However, if search engines exhibit non-monotonicity, the predictability of the results is greatly reduced.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This dissertation aims at providing a better search experience for Web users. Problems arise with the weaknesses of the current keyword-based search engines. The major search engines do not disambiguate homonymous search terms. The returned results contain mingled information of all homonyms and typically contain long list of pages of hits. To improve the Web search process with homonymous terms, the Ontology-Supported Web Search (OSWS) System was developed. The system clearly categorizes and disambiguates homonymous searches. The ontology used in the system was built from DBpedia and Facebook, in addition to using the suggested completions mined from Google.

To control the size of the returned Web results, to be neither too overwhelming nor too limited, a prediction mechanism has been developed. The prediction model was built based on the frequency of the search terms supplied to three common search engines, Google, Yahoo! Search and Bing. The study also evaluated how well the mechanism predicted the performance of the three search engines. During the study, it was found that all three search engines show non-monotonic behavior for negative search terms. Research results were presented, concerning the non-monotonicity of the three search engines for both positive and negative search terms.

Chapter 2 described an algorithm and its implementation in the OSWS System that improves previous work [17] on ontology-supported Web search. OSWS provides a method that generates better (than existing search engines) suggested completions of user

search terms when those search terms refer to homonyms. The interface clearly separates between suggested completions for homonymous concepts that fit the partial search term that a user has already typed in. OSWS uses an ontology to derive the disambiguated search terms and suggested search completions based on the knowledge about famous people in the ontology. Furthermore, suggested completions in the OSWS interface may contain positive and negative search words. OSWS allows users to include "negative search terms" in suggested completions, which further refine searches by negating search queries with information known not to apply to a given instance.

In order to improve the search experience of Web users, while discriminating between different senses of a homonymous term, a new interface has been developed for the OSWS System improving it in two ways. In the first stage, the system divides the snippet display into vertical panels to visually separate the results for the different homonyms. When the user moves the mouse down to one of the suggested completions and hovers there, the processing enters the second stage. For every suggested completion the user points to, the system instantly shows the result snippets of the suggestion he is hovering over. This improves the Google Instant feature. Currently Google shows instant results only for the first suggested completion. These two new features help the user acquire a deeper understanding of the suggested terms and to enjoy a better search experience than provided by today's search engines, while minimizing the number of actions she has to perform.

The current ontology is queried using special-purpose Java code. SPARQL is considered to be used in the future, as the ontology grows in size and complexity.

In *future work*, it is also planned to collect user feedback and to perform a formal evaluation study of the new features, e.g., using a tool such as Morae™. In order to test the usability of the OSWS System in improving the user search experience, future work will include conducting a user study to compare the OSWS System with the major search engines, such as Google, Yahoo and Bing, in aspects of usability and user satisfaction.

In [105], an evaluation study was conducted in an undergraduate course [105]. It is planned to involve 20 to 30 university students and scholars to participate in the user study. Each participant will be asked to perform Web searches using the same sets of queries on the OSWS System and on one of the major commercial search engines, depending on the user's preference.

Each participant will perform a set of 20 queries of his or her choice. Each query should consist of the name of a famous person. Based on such experiments, a post task questionnaire will be filled out by all users. It will focus on the usability and user satisfaction with the new features incorporated into the OSWS System, including the search continuation interface for disambiguating the homonymous search terms, user control of searching with negative terms, parallel result display and the instant feature.

Participant preferences will be obtained via questionnaires using Likert scales [106]. Most Likert scales have either 5, 7, or 9 degrees to choose among; odd numbers make it clear what the central or neutral choice is [107]. If comparing a new interface against one that is known already to have strong positive reactions, a wider scale allows for participants to clearly indicate a preference above and beyond what is already familiar and available [107].

The questionnaire will be designed based on a usability testing tool called Questionnaire for User Interaction Satisfaction (QUIS) [108]. QUIS is a measurement tool for evaluating computer users' subjective satisfaction with the human-computer interface [108]. Most QUIS-based questionnaires are arranged in a hierarchical layout: they start with a demographic questionnaire, which aims to determine user background information such as level of computer literacy. This is followed by measures of overall reaction towards the system. Finally, there are several specific interface sections.

The questionnaire will be adapted from QUIS and it will consist of:

(1) A background information section with questions relating to experience of using search engines;

(2) An overall-reaction section with different measures, such as level of satisfaction of using the OSWS System and the usability of disambiguating homonymous search terms.

(3) A section on the measurement of user satisfaction with the key features in the OSWS System; and

(4) A comments section which allows participants to provide comments and feedback that are related to possible areas of improvement of the OSWS System.

In part (2) above, participants will be asked to give their responses for both the OSWS System and the preferred commercial search engine, for each question being asked, so that the answers can be compared.

Chapter 3 reviewed the old knowledge base used in the OSWS System and described a new method for building an ontology of famous people by using search suggestions retrieved from Google, together with information extracted from DBpedia and YAGO. DBpedia is a huge public resource, but suffers from inconsistencies. The DBpedia ontology contains well-structured and consistent data but is very limited and

covers only a small portion of the domain. Thus, the person hierarchy was kept from the DBpedia ontology and information about the famous people was extracted from DBpedia.

The A-List of famous people was mined from Google's suggested completions. Various methods have been applied to clean the extracted DBpedia information, in order to consistently integrate the new knowledge into the ontology. This ontology was integrated into the Ontology-Supported Web Search System, which provides disambiguated search suggestions based on the ontology. A prototype expansion system was also developed for *dynamically* expanding the content of the ontology at run time, based on user-input queries, resulting in the D-OSWS System.

For the current version of this new ontology, the DBpedia person hierarchy was used, which is shallow but provides reasonable granularity for search suggestions. However, while comparing YAGO types to DBpedia ontology classes, it was found that there were quite a few missing classes in the DBpedia person hierarchy. For example, movie directors are grouped into the *Actor* class. Martin Scorsese is classified as an actor instead of a director.

Future work involves further refining the class hierarchy to provide better search results and exploring the use of new sources of data to include within the ontology. By further analyzing class names from other sources, it will be possible to provide better search suggestions in a number of cases. Additionally, research will continue on improving the domain coverage.

One problem has been recognized, that DBpedia is a (largely) static data source, when the goal is to keep up with a dynamically changing search environment. While DBpedia is working on a live extraction system [34, 109], the DBpedia ontology is

presently updated only twice per year. If the research relies solely on DBpedia as a data source, it may be not possible to find up-to-date information on certain instances in the ontology, and it is likely that some new instances will be missed entirely. It is planned to use new, more frequently updated, data sources to augment the OSWS ontology with fresh data, such as online news-feeds. By including new sources it will be possible to provide up-to-date and relevant search results to the end-users. *Future work* may also include expanding the OSWS System to perform what is called "Search What I Mean" (SWIM) queries, which will return results for what the system believes the user intends to search for. Lastly, work will continue with the B-List and possibly parts of the C-List.

Chapter 4 has presented the process of mining Facebook as a secondary resource to enhance the OSWS ontology. The study focused on the 2,564 names that exist in the A-List but could not be found in DBpedia, which were referred as the "reduced-A-List." Passing them to Facebook, it was possible to find 954 names that have a Facebook public page and are classified as person. Since the OSWS System is about *famous* people, the pages that have fewer than 300 fans were disregarded. A series of data extraction and data cleaning steps were done to mine the Facebook public pages of the selected people. The standardized data was then mapped to the OSWS ontology described in Chapter 3.

The process discussed in Chapter 4 involves data analysis and manual checking in specific steps. Having the "reduced-A-List" analyzed and successfully mapped to the ontology, mapping the people in the B-List and C-List would be a mostly automatic process. *Future work* involves enhancing the OSWS ontology by adding the people in the B-List and the C-List, who do not exist in DBpedia.

Chapter 5 has motivated and described a method for predicting hit count estimates based on the frequency of the search terms supplied to three common search engines, Google, Yahoo! Search and Bing. Frequencies were taken from a corpus of the English language. Second degree correlation functions were derived, based on random samples taken from the corpus. The derived regression functions were then used for the purpose of predicting the hit count estimates. Due to the varying behavior of the search engines, depending on the number of search terms passed to them, the study has derived separate correlation functions for 30 different cases of search terms of varying length, with and without negative terms. The experiments indicated that the predictions made for the samples were sufficiently close to the hit count estimates returned by the search engines to make them useful. Among the three search engines, Bing gives better results (closer correlation between the term frequencies and the numbers of hits) than Google and Yahoo! Search.

The Google Ajax Search API, Yahoo! Boss API and Bing API were used to fetch the hit count estimates of the three search engines. The Google Custom Search API is the substitute of the Google Ajax Search API. However, it provides only up to 100 free queries per day [83], which has caused great inconvenience in continuation of this study.

Future research includes extending work on hit count estimates in several directions. For common phrases, the frequency (co-occurrence) of a multi-word term is often not the same as what would be predicted by the component word frequencies. Thus, one can expect more Michael Jackson hits than predicted by Michael and Jackson separately, in contrast to Alfred Jackson hits. In the future, the frequency of the dependent terms will be updated using knowledge retrieved from relevant ontologies.

Secondly, *future research* involves adding specialized word frequency lists to the database, such as US Census first name and last name frequency information. This will reduce the number of words for which the prediction model currently does not have any frequency estimates. Besides, additional data sources will be looked into that cover newly-coined, commonly-used words, e.g., 'facebooking'.

The search engine hit count estimates are computed using sampling algorithms, and thus are expected to vary significantly. Rather than choosing the 5000 most frequent words, a random sample may be more interesting in terms of results.

The *future study* will also consider the Corpus of the American Contemporary English (COCA) [110] as a data set. Most importantly, it is considered to integrate the thirty prediction models into fewer, ideally one for each search engine, to simplify the modeling and increase the usability of the prediction model. Moreover, since the Web is constantly growing, the prediction models should be updated periodically. Another possible extension of this part of the work could be to expose a REST API that provides the predicted hit count estimates for given search queries.

One interesting research topic would be to analyze the correlation between the popularity of a Wikipedia page and the search engines' hit counts for this page. Wikipedia has the advantage of multi-domain and multi-language coverage. Thus, it would be interesting to investigate the correlation between page popularity and search engine results.

Chapter 6 studied the effect of adding negative and positive search words to existing queries. It was shown that when between one and five negative search words were added to a single positive search word, the observed hit count estimates did not

behave monotonically for many cases. For example, for one positive search word, Google rarely behaves correctly when a sequence of negative search words is appended. Compared to Google, Yahoo! Search and Bing behave much better in following the announced behavior for adding negative terms. In this study, Yahoo! Search shows the best behavior, with 98.8% of the experiments exhibiting the correct monotonic behavior. The research also investigated the effect of adding negative search words after two positive words and adding only positive search words in the queries. The three search engines show better monotonicity results in those cases, but still do not conform to the documented behavior.

Future work continues on investigating the monotonic behavior of the popular search engines. For example, Google ignores common words and characters such as where, the, how, and includes synonyms automatically in its processing. However, adding a plus operator (+) in front of a word would indicate to Google that it should search with the exact word the user included in his query [111]. In the future, it is planned to analyze and compare the monotonicity of the search engines with and without using the plus operator.

REFERENCES

[1] D.R. Radev, W. Fan, and Z. Zhang, "WebInEssence: A Personalized Web-Based Multi-Document Summarization and Recommendation Syatem", NAACL Workshop on Automatic Summarization, Pittsburgh, PA, 2001.

[2] Google Query Suggestion, http://www.google.com/support/websearch/bin/answer.py?hl =en&answer=106230, retrieved 12/01/2011.

[3] E. Al-Masri, Q.H. Mahmoud, "Discovering Web Services in Search Engines," IEEE Internet Computing, issue 3, pp. 74-77, 2008.

[4] S.R. Lawrence, "Personalization of Web Search Results Using Term, Category, and Link-Based User Profiles," United States Patent Appl. 20100228715. Kind Code: A1, 2010.

[5] T. Tian, J. Geller, S.A. Chun, "Predicting Web Search Hit Counts," 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp. 162-166, Toronto, Canada, 2010.

[6] T. Tian, J. Geller, S.A. Chun, "Improving Web Search Results for Homonyms by Suggesting Completions from an Ontology," 2nd International Workshop on Semantic Web Information Management (SWIM). Lecture Notes in Computer Science, pp. 175-186. Vienna, Austria, 2010.

[7] F. Radlinski, M. Szummer, N. Craswell, "Inferring Query Intent from Reformulations and Clicks," WWW 2010, Raleigh, North Carolina, USA, 2010.

[8] M. Henzinger, "Search Technologies for the Internet," Science, vol. 317, no. 5837, pp. 468-471, 2007.

[9] R. Capra, G. Marchionini, J. Velasco-Martin and K. Muller, "Tools-at-hand and Learning in Multi-session, Collaborative Search,": 28th International Conference on Human Factors in Computer Systems (CHI'10) Atlanta, Georgia, 2010.

[10] L. Yuen, M. Chang, Y.K. Lai, C.K. Pool, "Excalibur: A Personalized Meta Search Engine," 28th Annual International Computer Science Software and Applications Conference (COMPSAC'04), vol. 2, pp. 49-50, September 2004.

[11] M. Thelwall, "Extracting Accurate and Complete Results from Search Engines: Case Study Windows Live," Journal of the American Society for Information Science and Technology, vol. 59, issue 1, pp. 38-50, 2007.

[12] E.T. Jepsen, P. Seiden, P. Ingwersen, L. Björneborn, P. Borlund, "Characteristics of Scientific Web Publications: Preliminary Data Gathering and Analysis," Journal of the American Society for Information Science and Technology, vol. 55, issue 14, pp. 1239-1249, 2004.

[13] T. Joachims and F. Radlinski, “Search Engines that learn from Implicit Feedback”, IEEE Computer, IEEE Computer Society, vol. 40, issue 8, pp. 34-40, 2007.

[14] iProspect Search Engine User Behavior Study, http://www.iprospect.com/ premiumPDFs/WhitePaper_2006_SearchEngineUserBehavior.pdf, 2006, retrieved 12/01/2011.

[15] F. Baader, “Description Logics,” Reasoning Web: Semantic Technologies for Information Systems, 5th International Summer School, vol. 5689, pp. 1–39, Lecture Notes in Computer Science,. Springer Verlag, 2009.

[16] DBpedia, http://dbpedia.org/About, retrieved 12/01/2011.

[17] Y. An, S. Chun, K. Huang, J. Geller, “ Enriching Ontology for Deep Web Search,” DEXA, Lecture Notes in Computer Science, pp. 73-80, Turin, Italy, 2008.

[18] Y. An, J. Geller, Y. Wu, S.A. Chun, “Semantic Deep Web: Automatic Attribute Extraction from the Deep Web Data Sources,” 2007 ACM Symposium on Applied computing (ACM-SAC), pp. 1667-1672, Seoul, Korea, 2007.

[19] G. Fu, C.H. Jones, A.I. Abdelmoty, “Ontology-based Spatial Query Expansion in Information Retrieval,” ODBASE: OTM Confederated International Conferences, Agia Napa, Cyprus, 2005.

[20] R. Navigli R, P. Velardi, “An Analysis of Ontology-based Query Expansion Strategies,” Workshop on Adaptive Text Extraction and Mining (ATEM 2003), 14th European Conference on Machine Learning (ECML 2003), Cavtat- Dubrovnik, Croatia, 2003.

[21] A. Andreou, “Ontologies and Query Expansion,” M.S. thesis, School of Informatics, Edinburgh Univ., Edinburgh, UK, 2005.

[22] Y. An, S.A. Chun, K. Huang, J. Geller, “Assessment for Ontology-Supported Deep Web Search,” 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and Eservices, IEEE Computer Society, pp. 382-388, Washington D.C, 2008.

[23] D.L. McGuinness, “Ontologies Come of Age,” In D. Fensel, J. Hendler, H. Lieberman, W. Wahlster (eds) Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, MIT Press, 2003.

[24] A. Uyar, “Investigation of the Accuracy of Search Engine Hit Counts,” Journal of Information Science , vol. 35, issue 4, pp. 469-480, 2009.

[25] K. Sugiyama, M. Okumura, “Personal Name Disambiguation in Web Search Results Based on a Semi-supervised Clustering Approach”, Lecture Notes in Computer Science, vol. 4822, pp. 250-256, 2007.

[26] C. Chen, J. Hu, H. Wang, “Clustering Technique in Multi-Document Personal Name Disambiguation”, ACL-IJCNLP 2009 Student Research Workshop, Suntec, Singapore, August 2009.

[27] D. Rao, J. Garera, D. Yarowsky, "JHU1: An Unsupervised Approach to Person Name Disambiguation using Web Snippets", 4th International Workshop on Semantic Evaluations, Prague, June 2007.

[28] B. Fazzinga, T. Lukasiewicz, "Semantic Search on the Web," Semantic Web – Interoperability, Usability, Applicability, vol. 1, pp. 1-7, 2010.

[29] R.V. Guha, R. McCool, E. Miller, "Semantic Search," WWW'03, pp. 700-709, Budapest, Hungary, 2003.

[30] G. Cheng, W. Ge, Y. Qu, "Falcons: Searching and Browsing Entities on the Semantic Web," WWW'08, pp. 1101-1102, Beijing, China, 2008.

[31] A. Harth, A. Hogan, R. Delbru, J. Umbrich, S. O'Riain, S. Decker, "SWSE: Answer before Links!," Semantic Web Challenge, CEUR Workshop, 2007.

[32] G. Tummarello, R. Cyganiak, M. Ctasta, S. Danielczyk, R. Delbru, S. Decker, "Sig.Ma: Live Views on the Web of Data," WWW 2010, pp. 1301-1304, Raleigh, NC, USA, 2010.

[33] Z. B. Yossef, M. Gurevich, "Mining search engine query logs via suggestion sampling," VLDB Endowment, Auckland, New Zealand, vol. 1, issue 1, pp. 54-65, 2008.

[34] C. Bizer et al., "DBpedia - a Crystallization Point for the Web of Data," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 7, issue 3, pp. 154-165, 2009.

[35] M. Kalender, J. Dang, S. Uskudarli, UNIpedia: "A Unified Ontological Knowledge Platform for Semantic Content Tagging and Search," ICSC '10 2010 IEEE Fourth International Conference on Semantic Computing, Washington DC, pp. 293-298, 2010.

[36] D.M. Boyd, N.B. Ellison, "Social Network Sites: Definition, History, and Scholarship," Journal of Computer-Mediated Communication, vol. 13, issue 1, pp. 210-230, 2007.

[37] M. Thelwall, D. Wilkinson, S. Uppal, "Data Mining Emotion in Social Network Communication: Gender Differences in MySpace," Journal of the American Society for Information Science and Technology, vol. 61, issue 1, pp. 190-199, 2010.

[38] H. Chu, D. Deng, J.H. Park, "Live Data Mining Concerning Social Networking Forensics Based on a Facebook Session Through Aggregation of Social Data," IEEE Journal of Selected Areas in Communications, vol. 29, issue 7, pp. 1368-1376, 2011.

[39] K. Xu, S.S. Liao, Y. Song, L. Liu, "Mining User Opinions in Social Network Webs," The Fourth China Summer Workshop on Information Management, Wuhan, China, 2010.

[40] I. Guy, M. Jacovi, E. Shahar, N. Meshulam, V. Soroka, "Harvesting with SONAR - The Value of Aggregating Social Network Information," CHI, Florence, Italy, 2008.

[41] Y. Matsuo, J. Mori, M. Hamasaki, "POLYPHONET: An Advanced Social Network Extraction System from the Web," International World Wide Web Conference (WWW), Edinburgh, Scotland, 2006.

[42] M. Thelwall, "Introduction to Webometrics: Quantitative Web Research for the Social Sciences", Synthesis Lectures on Information Concepts, Retrieval, and Services, vol. 1, no. 1, pp. 1-116, 2009,

[44] R. Cilibrasi, P. Vitanyi, "The Google Similarity Distance", IEEE Trans. Knowledge and Data Engineering, vol. 19, issue 3, pp. 370-383, 2007.

[44] D. Sánchez, M. Batet M, A. Valls, "Web-Based Semantic Similarity: An Evaluation in the Biomedical Domain," International Journal of Software and Informatics, vol. 4, issue 1, pp. 39-52, 2010.

[45] R. Mirizzi, A. Ragone, T. Noia, E. Sciascio, "Semantic Tags Generation and Retrieval for Online Advertising," 19th ACM International Conference on Information and Knowledge Management (CIKM), pp. 1089-1098, Toronto, Canada, 2010.

[46] R. Mirizzi, A. Ragone, T. Noia, E. Sciascio, "Ranking the Linked Data: the Case of DBpedia," 10th International Conference on Web Engineering (ICWE), pp. 337-354,Vienna, Austria, 2010.

[47] Z.B. Yossef, M. Gurevich, "Random Sampling from a Search Engine's Index", 15th International World Wide Web Conference (WWW), Edinburgh, Scotland, 2006.

[48] British National Corpus, http://www.natcorp.ox.ac.uk/, retrieved 12/01/2011.

[49] Y. Matsuo, H. Tomobe, T. Nishimura, "Robust Estimation of Google Counts for Social Network Extraction", WWW 2007, Banff, Canada, May 2007.

[50] Facebook, www.facebook.com, retrieved 12/01/2011.

[51] T. Tian, J. Geller, S.A. Chun, "Enhancing the Interface for Ontology-Supported Homonym Search," CAiSe'11 Workshop: 1st International Workshop on Semantic Web Search (SSW), Lecture Notes in Computer Science, London, UK, 2011.

[52] Yahoo Search Assistant, http://tools.search.yahoo.com/newsearch/searchassist.html, retrieved 12/01/2011.

[53] Bing Search Suggestions, http://onlinehelp.microsoft.com/en-us/bing/ff808490.aspx, retrieved 12/01/2011.

[54] Y. Ke, L. Deng, W. Ng, D. Lee, "Web Dynamics and their Ramifications for the Development of Web Search Engines," Computer Networks: The International Journal of Computer and Telecommunications Networking – Web Dynamics, vol. 50, issue 10, pp. 1430-1447, 2006.

[55] Google Instant, http://www.google.com/instant/, retrieved 12/01/2011.

[56] T.B. Lee, J. Hendler, O. Lassila, "The Semantic Web", Scientific American Magazine, May 2001.

[57] L. Ding, T. Finin, A. Joshi, R. Pan, R.S. Cost, "Swoogle: A Search and Metadata Engine for the Semantic Web", thirteenth ACM international conference on Information and knowledge management, pp. 652-659, ACM Press, 2004.

[58] Ontology Design Patterns (ODP), http://ontologydesignpatterns.org/wiki/Main_Page, retrieved 10/15/2010.

[59] Open Biological and Biomedical Ontologies (OBO), http://www.obofoundry.org/, retrieved 12/01/2011.

[60] L. Niles, A. Pease, "Towards a standard upper ontology," International Conference on Formal Ontology in Information System, pp. 2-9, ACM, New York, 2001.

[61] J.F. Sowa, "Knowledge Representation: Logical, Philosophical, and Computational Foundations," Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[62] C. Ochs, T. Tian, J. Geller S.A. Chun, "Google Knows Who is Famous Today: Building an Ontology from Search Engine Knowledge and DBpedia," 5th IEEE International Conference on Semantic Computing (ICSC), Palo Alto, CA, 2011.

[63] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, "DBpedia: a Nucleus for a Web of Open Data," ISWC'07/ASWC'07 6th International Semantic Web and 2nd Asian Conference on Asian Semantic Web, Busan, Korea, pp. 722-735, 2007.

[64] DBpedia Disadvantage, http://wiki.dbpedia.org/UseCases, retrieved 12/01/2012.

[65] DBpedia SPARQL, http://dbpedia.org/sparql, retrieved 12/01/2012.

[66] Google Autocomplete SOAP API, http://docs.jquery.com/UI/Autocomplete, retrieved 12/01/2012.

[67] Government Census Data, http://www.census.gov/, retrieved 12/01/2012.

[68] WordNet, http://wordnet.princeton.edu/, retrieved 12/01/2012.

[69] F. M. Suchanek, G. Kasneci, G. Weikum, "YAGO: a Core of Semantic Knowledge," WWW '07 16th International Conference on WWW, pp. 697-706, Banff, Alberta, Canada, 2007.

[70] J. Euzenat, P. Shvaiko, "Ontology Matching," Springer-Verlag, Berlin, Heidelberg, 2007.

[71] About Facebook Pages, http://www.facebook.com/help/pages/admin, retrieved 12/01/2011.

[72] Facebook Graph API, http://developers.facebook.com/docs/reference/api/, retrieved 12/01/2011.

[73] Inside Facebook Pages, http://www.sysomos.com/insidefacebook/, retrieved 12/01/2011.

[74] T. McCorkindale, "Can You See the Writing on My Wall? A Content Analysis of the Fortune 50's Facebook Social Networking Sites," Public Relations Journal, vol. 4, no. 3, pp. 1-10, 2010.

[75] Facebook Pages, http://developers.facebook.com/docs/reference/api/page/, retrieved 12/01/2011.

[76] Stands4 API, http://www.abbreviations.com/api.asp, retrieved 12/01/2011.

[77] T. Tian, S.A. Chun, J. Geller, "A Prediction Model for Web Search Hit Counts Using Word Frequencies," Journal of Information Science, vol. 37, issue 5, pp. 462-475, Sage Publishing Co., 2011.

[78] P. Nakov, M. Hearst, "A study of Using Search Engine Page Hits as a Proxy for n-gram Frequencies", Recent Advances in Natural Language Processing, Borovets, Bulgaria, September 2005.

[79] J. Pollard, "Google result counts are a meaningless metric", http://homepage.ntlworld.com/jonathan.deboynepollard/FGA/google-result-counts-are-a-meaningless-metric.html, retrieved 12/01/2011.

[80] S. Brin, L. Page, "The Anatomy of a Large-scale Hypertextual Web Search Engine", Computer Networks, vol. 30, issue 1-7, pp. 107-117,1998.

[81] T. Funahashi, H. Yamana, "Reliability Verification of Search Engines' Hit Counts", 1st International Workshop on Quality in Web Engineering, Vienna, Austria, July 2010.

[82] Google Search API, http://code.google.com/apis/websearch/, retrieved 12/01/2012.

[83] Google Custom Search API, http://code.google.com/apis/websearch/, retrieved 12/01/2012.

[84] Google SOAP Search API, http://code.google.com/apis/soapsearch/, retrieved 12/01/2012.

[85] P. Mayr, F. Tosques, "Google Web APIs: An Instrument for Webometric Analyses", ISSI 2005 Conference, Stockholm, Sweden, July 2005.

[86] A. Kilgarriff, "Googleology is Bad Science", Computational Linguistics, vol. 33, issue 1, pp. 147-151, 2007.

[87] R. Gligorov, W.T. Kate, Z. Aleksovski, F. V Harmelen, "Using Google Distance to Weight Approximate Ontology Matches", 16th International World Wide WebConference, pp. 767-776, Banff, Alberta Canada, May 2007.

[88] F. Keller, M. Lapata, "Using the Web to Obtain Frequencies for Unseen Bigrams", Computational Linguistics, vol. 29, issue 3, pp. 459-484, 2003.

[89] Brown Corpus, http://www.edict.com.hk/lexiconindex/frequencylists/words2000.htm, http://www.edict.com.hk/lexiconindex/frequencylists/words2-5k.htm, retrieved 03/21/2010.

[90] Stop words, http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words, retrieved 12/01/2012.

[91] G. Zipf, Selective Studies and the Principle of Relative Frequency in Language, Cambridge Press, Cambridge, Mass, 1932.

[92] G. Zipf, Human Behavior and the Principle of Least-Effort, Cambridge Press, Cambridge, Mass, 1949, Addison-Wesley, 1965.

[93] G. Zipf, The Psycho-biology of Language: An Introduction to Dynamic Philology, Houghton-Mifflin Company, 1935, MIT Press, 1965.

[94] P. Pantel, D. Lin, "A Statistical Corpus-Based Term Extractor", 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, LNCS, vol. 2056, pp. 36-46, 2001.

[95] M. Srikanth, "Exploiting Query Features in Language Modeling Approach for Information Retrieval", PhD thesis, State University of New York at Buffalo, 2004.

[96] W. Szpankowski, "Digital Data Structures and Order Statistics," LNCS: Proceedings of Workshop WADS'89, pp. 206-217, Ottawa, Canada, 1989.

[97] S. Goldwater, T.L. Griffiths, M. Johnson, "Distributional Cues to Word Boundaries: Context is Important," 31st Annual Boston University Conference on Language Development, Cascadilla Press, Somerville, MA, 2007.

[98] D. Miller, T. Leek, R.M. Schwartz, "A Hidden Markov Model Information Retrieval System," SIGIR'99, pp. 214-221, Berkley, California, 1999.

[99] F. Song F, W.B. Croft, "A General Language Model for Information Retrieval," SIGIR'99, pp. 316-321, Berkeley, California, 1999.

[100] R.R. Wilcox, Trimmed Means, In B.S. Everitt, D.C. Howell, Encyclopedia of Statistics in Behavioral Science, Wiley, Chichester, England, 2005.

[101] I.H. Witten, E. Frank, Data mining: Practical machine learning tools and techniques (second edition), Morgan Kaufmann, San Francisco, CA, 2005.

[102] Google Search Basics: Basic Search Help, http://www.google.com/support/websearch/bin/answer.py?hl=en&answer=134479, retrieved 12/01/2012.

[103] Yahoo Help: Search Tips, http://help.yahoo.com/l/us/yahoo/search/narrowyoursearch/basics-04.html;_ylt=AggxCl0pmWi9tjzBuKskoYh6YXhG, retrieved 12/01/2012.

[104] Bing Help: Search Tips, http://onlinehelp.microsoft.com/en-us/bing/ff808438.aspx, retrieved 12/01/2012.

[105]M. Chau, C.H. Wong, "Designing the User Interface and Functions of a Search Engine Development Tool," Decision Support Systems, vol. 28, issue 2, pp. 369-382, 2010.

[106]R. Likert: "A Technique for the Measurement of Attitudes," Archives of Psychology, vol. 22, no. 140, pp. 1-55, 1932.

[107]M. Hearst: Search User Interfaces, Cambridge University Press, 2009.

[108]J. Chin, V. Diehl, K. Norman: "Development of an Instrument Measuring User Satisfaction of the Human–Computer Interface," Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Washington, D.C., USA, 1988, pp. 213–218.

[109]DBpedia Next Step, http://wiki.dbpedia.org/NextSteps, retrieved 12/01/2012.

[110]Corpus of the American Contemporary English (COCA), http://corpus.byu.edu/coca/.

[111]Google Plus Operator, http://www.google.com/support/websearch/bin/answer.py?answer=136861, retrieved 12/01/2012.