SEMANTIC RELATEDNESS FOR EVALUATION OF COURSE EQUIVALENCIES

BY

BEIBEI YANG
B.S., JIANGSU POLYTECHNIC UNIVERSITY, CHINA (2003)
M.S., UNIVERSITY OF MASSACHUSETTS LOWELL (2009)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL

Signature of Author: Beibei Yang Date: 7/23/2012

Signature of Dissertation Chair: Jesse Heines
Jesse M. Heines, Ed.D.

Signatures of Other Dissertation Committee Members

Committee Member Signature: Karen M. Daniels
Karen M. Daniels, Ph.D.

Committee Member Signature: Georges Grinstein
Georges G. Grinstein, Ph.D.

Committee Member Signature: Lisa N. Michaud
Lisa N. Michaud, Ph.D.

SEMANTIC RELATEDNESS FOR EVALUATION OF COURSE EQUIVALENCIES

BY

BEIBEI YANG

ABSTRACT OF A DISSERTATION SUBMITTED TO THE FACULTY OF THE
DEPARTMENT OF COMPUTER SCIENCE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
COMPUTER SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL
2012

Dissertation Supervisor: Jesse M. Heines, Ed.D.
Professor, Department of Computer Science

To my parents,

and

to Andy.

ACKNOWLEDGMENTS

This work would not be possible without the help of many people. First and foremost I want to express my deepest gratitude to my advisor at University of Massachusetts Lowell (UML), Jesse Heines. I thank him for having faith in me, for helping me develop ideas, and for working patiently on my writing. He offered me mentorship toward becoming a great researcher as well as a great person. He made it top priority to help his students to excel, and was always there to offer the guidance when it's needed, regardless of how busy he was. He taught me the difference between good writing and great writing. He taught me how to distill important and challenging problems to work on, and reminded me of the right path when I proceeded too far. He encouraged me to take control of my research, while continuously shedding the light when I was stuck in dark corners.

I also owe a huge debt of gratitude to my other committee members: Karen Daniels, Georges Grinstein, and Lisa Michaud, who offered extensive support and valuable feedback which guided me to improve the research.

I can't thank Saif Mohammad enough for being incredibly supportive and helping move this research in interesting directions. It was a great honor to have him as my mentor at the NAACL conference. Our long discussion at the conference guided improvements to the work. I would like to express my appreciation to Yuhua Li, Dongqiang Yang, and Mehran Sahami for correspondences which were a huge help at the early stages of my research.

I am grateful to the UML Department of Computer Science. All faculty members, staff and students made my 5-year study in this department a wonderful journey. In particular, Zheng Fang helped me set up some of the Wikipedia data. Input from Brendan Reilly and Angelo Gamarra helped me set up the human judgment data set. I feel fortunate to have the opportunity to work with a group of brilliant people in

the CNIS lab at UML. I enjoyed working together with former and present CNIS lab members, for day and late night lab activities.

Finally, I would like to thank my parents, my husband, and the rest of my family for all the support, care, happiness, love, and everything else.

ABSTRACT

SEMANTIC RELATEDNESS FOR EVALUATION OF
COURSE EQUIVALENCIES

Beibei Yang

Semantic relatedness, or its inverse, semantic distance, measures the degree of closeness between two pieces of text determined by their meaning. Related work typically measures semantics based on a sparse knowledge base such as WordNet or Cyc that requires intensive manual efforts to build and maintain. Other work is based on a corpus such as the Brown corpus, or more recently, Wikipedia.

This dissertation proposes two approaches to applying semantic relatedness to the problem of suggesting transfer course equivalencies. Two course descriptions are given as input to feed the proposed algorithms, which output a value that can be used to help determine if the courses are equivalent. The first proposed approach uses traditional knowledge sources such as WordNet and corpora for courses from multiple fields of study. The second approach uses Wikipedia, the openly-editable encyclopedia, and it focuses on courses from a technical field such as Computer Science.

This work shows that it is promising to adapt semantic relatedness to the education field for matching equivalencies between transfer courses. A semantic relatedness measure using traditional knowledge sources such as WordNet performs relatively well on non-technical courses. However, due to the "knowledge acquisition bottleneck," such a resource is not ideal for technical courses, which use an extensive and growing set of technical terms. To address the problem, this work proposes a Wikipedia-based approach which is later shown to be more correlated to human judgment compared to previous work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

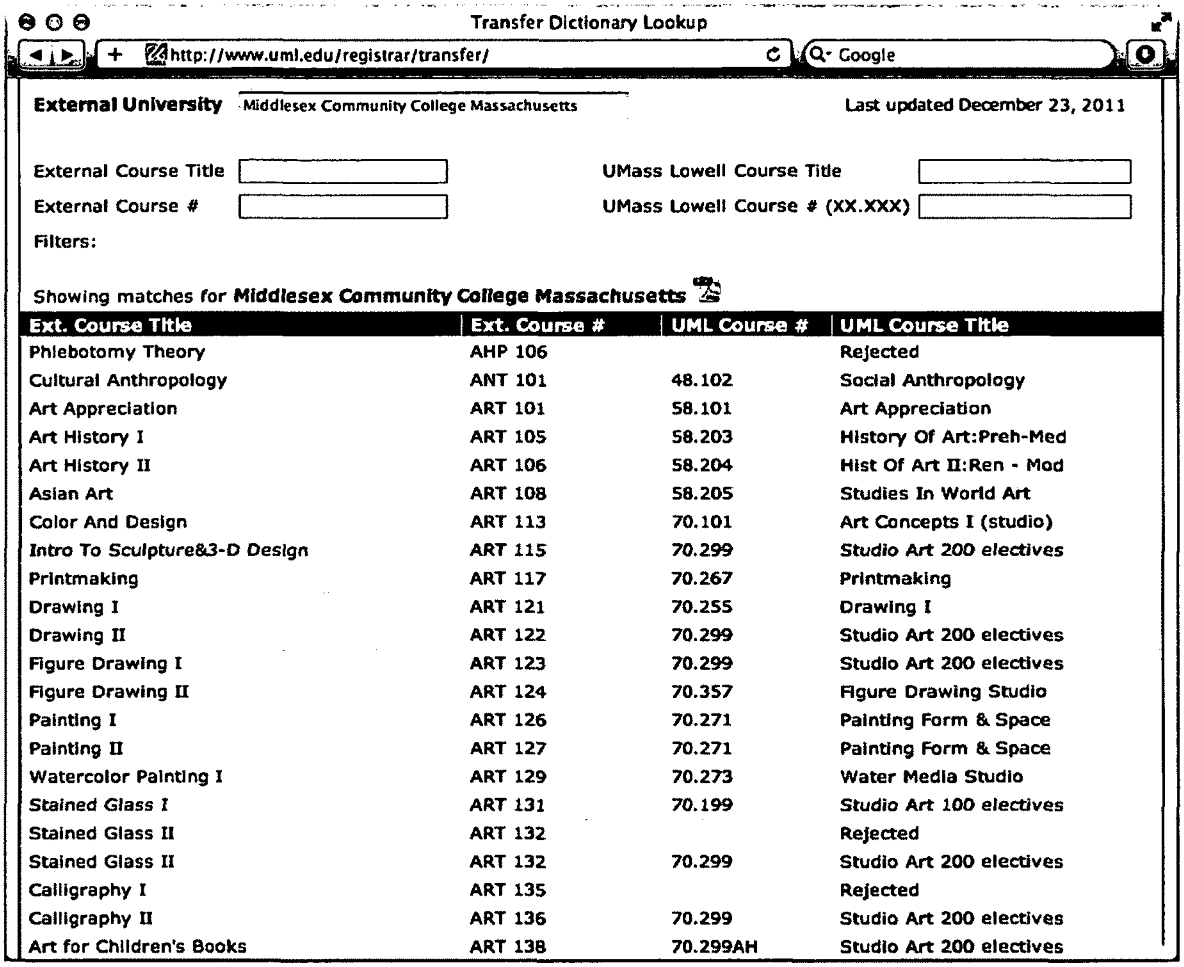
CHAPTER 1

INTRODUCTION

1.1 The Problem

Many *natural language processing* (NLP) techniques have been adapted to the education field for building systems such as automated scoring, intelligent tutoring, and learner cognition. Few, however, address the identification of transfer course equivalencies. A report released by the National Center for Education Statistics in 2005 shows that for students who attained their bachelor's degrees in 1999–2000, 59% attended more than one institution during their undergraduate careers and 32.1% transferred at least once [52]. A recent study [49] conducted by the National Association for College Admission Counseling further states that 1/3 of US college students transfer to another institution.

Each year the University of Massachusetts Lowell (UML) accepts hundreds of transferring students. Courses taken at students' previous educational institutions must be evaluated by UML for transfer credit. Course descriptions are usually short paragraphs of fewer than 200 words. To determine whether an incoming course can be transferred, the undergraduate and graduate transfer coordinators from each department must manually compare its course description to the courses offered at UML. This process is labor-intensive and highly inefficient. There is a publicly available *course transfer dictionary* (Figure 1.1) which lists course numbers from hundreds of institutions and their equivalent courses at UML, but the data set is sparse, non-uniform, and always out of date. External institutions cancel courses, change course numbers, etc., and such information is virtually impossible to keep up to date in the transfer dictionary. Furthermore, the transfer dictionary does not list course descriptions. From our experience, course descriptions change over the years even when course numbers do not, and this of course affects equivalencies.



Transfer Dictionary Lookup

http://www.uml.edu/registrar/transfer/

External University Middlesex Community College Massachusetts — Last updated December 23, 2011

External Course Title

External Course #

UMass Lowell Course Title

UMass Lowell Course # (XX.XXX)

Filters:

Showing matches for **Middlesex Community College Massachusetts**

Ext. Course Title	Ext. Course #	UML Course #	UML Course Title
Phlebotomy Theory	AHP 106		Rejected
Cultural Anthropology	ANT 101	48.102	Social Anthropology
Art Appreciation	ART 101	58.101	Art Appreciation
Art History I	ART 105	58.203	History Of Art:Preh-Med
Art History II	ART 106	58.204	Hist Of Art II:Ren - Mod
Asian Art	ART 108	58.205	Studies In World Art
Color And Design	ART 113	70.101	Art Concepts I (studio)
Intro To Sculpture&3-D Design	ART 115	70.299	Studio Art 200 electives
Printmaking	ART 117	70.267	Printmaking
Drawing I	ART 121	70.255	Drawing I
Drawing II	ART 122	70.299	Studio Art 200 electives
Figure Drawing I	ART 123	70.299	Studio Art 200 electives
Figure Drawing II	ART 124	70.357	Figure Drawing Studio
Painting I	ART 126	70.271	Painting Form & Space
Painting II	ART 127	70.271	Painting Form & Space
Watercolor Painting I	ART 129	70.273	Water Media Studio
Stained Glass I	ART 131	70.199	Studio Art 100 electives
Stained Glass II	ART 132		Rejected
Stained Glass II	ART 132	70.299	Studio Art 200 electives
Calligraphy I	ART 135		Rejected
Calligraphy II	ART 136	70.299	Studio Art 200 electives
Art for Children's Books	ART 138	70.299AH	Studio Art 200 electives

Figure 1.1: UML's course transfer dictionary

This work proposes two approaches to automatically suggest course equivalencies by analyzing the course descriptions and comparing their semantic relatedness. The course descriptions are first pruned and unrelated contexts are removed. Given a course from another institution, the algorithm measures the relatedness of its description to descriptions in a list of courses offered at UML and suggests potentially equivalent courses. This work has two goals: (1) to assist transfer coordinators by suggesting equivalent courses within a reasonable amount of time on a standard laptop system, and (2) to explore new applications using semantic relatedness to move toward the Semantic Web [3], i.e., to turn existing resources into knowledge structures.

Each of the two proposed approaches is essentially a mapping function: $f : (C_1, C_2) \to n, n \in [0,1]$, where C_1 is a course from an external institution, and C_2 is a course offered at UML.

Each course description contains a *course title* and a *course abstract*. The course title consists of a few words that distinguish it from other courses within an institution. The course abstract is typically a short text passage.

Below are two course descriptions C_1 and C_2:

> C_1: "[**Analysis of Algorithms**] Discusses basic methods for designing and analyzing efficient algorithms emphasizing methods used in practice. Topics include sorting, searching, dynamic programming, greedy algorithms, advanced data structures, graph algorithms (shortest path, spanning trees, tree traversals), matrix operations, string matching, NP completeness."
>
> C_2: "[**Computing III**] Object-oriented programming. Classes, methods, polymorphism, inheritance. Object-oriented design. C++. UNIX. Ethical and social issues."

The output n of the mapping function is a real number between 0 and 1. A larger value of n indicates that C_1 and C_2 are more semantically related.

1.2 Knowledge Acquisition Bottleneck

Semantic relatedness measures that rely on a traditional knowledge source usually suffer the *knowledge acquisition bottleneck*. These knowledge source include, but are not limited to dictionaries (Section 2.1.1), thesauri (Section 2.1.2), WordNet (Section

2.1.3), Cyc (Section 2.1.4), and the British National Corpus (Section 2.2.2). The term *knowledge acquisition* originates from *expert systems* [26, 63]. *Knowledge acquisition* is the transfer and transformation of knowledge or expertise from the forms in which it is available in the world into forms that can be used by a knowledge system.

As previous research [26, 62] points out, *knowledge acquisition* experiences a few difficulties:

1. **Representation mismatch**: the difference between the way a human expert states knowledge and the way it is represented in the system.

2. **Knowledge inaccuracy**: the difficulty for human experts to describe knowledge in terms that are precise, complete, and consistent enough for use in a computer program.

3. **Coverage problem**: the difficulty of characterizing all of the relevant domain knowledge in a given representation system, even when the expert is able to correctly verbalize the knowledge.

4. **Maintenance trap**: the time required to maintain a knowledge source. As the knowledge in the knowledge source grows, so does the requirement for maintenance.

The *knowledge acquisition bottleneck* arises with the above difficulties. Knowledge must be acquired before anything can happen. Sources of knowledge are unreliable in that domain experts may not articulate their knowledge well and the knowledge they provide may be incomplete and even incorrect. Moreover, knowledge sources are difficult to build and representations of knowledge in a knowledge source may be complex.

1.3 Contributions

This thesis embodies several important contributions. It presents the problem of suggesting transfer course equivalencies. It proposes two semantic relatedness measures to tackle the problem. The first approach uses traditional knowledge sources to suggest course equivalencies from multiple majors, which is later shown to perform better on non-technical courses in fields such as art, philosophy, and history than on technical courses in fields such as computer science. The second focuses on technical courses using Wikipedia as the knowledge source. For these courses, both accuracy and correlation indicate that the second approach outperforms previous work.

The second approach shows that, although the rapid growth of Wikipedia makes the *knowledge acquisition bottleneck* less of a problem, it also makes it more challenging to parse such a huge resource in a reasonable amount of time. To address this issue, this work proposes a domain-specific semantic relatedness measure based on part of Wikipedia to suggest course equivalencies for course descriptions from a technical domain. This approach can be easily modified for other majors and even for other languages.

This work also presents a human judgment data set of course pairs from Computer Science. Future work can benefit from such a data set by computing correlation coefficients between this data set and the proposed relatedness measures.

1.4 Organization of the Thesis

The rest of the thesis is constructed as follows. Chapter 2 surveys some of the popular knowledge sources used in related work for measuring semantics. These knowledge sources are categorized into *lexicon*-based resources, corpus-based resources, and hybrid resources. Chapter 3 is an overview of related work on semantic relatedness and word sense disambiguation. Some of the semantic relatedness measures are based solely on lexicographic resources. Others are either based solely on corpora, or combine lexicons with corpora. Chapter 4 proposes a generic approach based on traditional resources such as WordNet and the Brown corpus to suggest equivalent courses from multiple fields of study. Chapter 5 proposes a domain-specific approach based on Wikipedia to suggest equivalent courses for a particular major. Finally, chapter 6 concludes the dissertation.

CHAPTER 2

POPULAR RESOURCES AS KNOWLEDGE BASES

Knowledge sources used by related literature for computation of semantics can be divided into three categories (as shown in Table 2.1). This chapter reviews some of the popular knowledge sources.

Type of Knowledge Sources	**Examples**
Lexicon-based resources	Dictionaries, Thesauri, WordNet, and Cyc
Corpus-based resources	Project Gutenberg, British National Corpus, and Penn Treebank
Hybrid resources	Wikipedia and Wiktionary

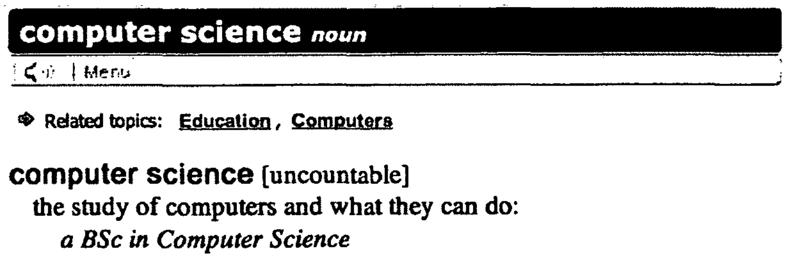
Table 2.1: Types of knowledge sources

2.1 Lexicon-based Resources

A traditional semantic relatedness measure uses one or more *lexicon*-based resources. These resources are usually manually created and maintained by small numbers of domain experts.

2.1.1 Dictionaries

A dictionary such as the Longman Dictionary of Contemporary English (LDOCE) provides definitions of words used in a natural language (Figure 2.1). Some related work has used the definitions in LDOCE as a clue to the semantic relatedness of words.



computer science *noun*

Menu

Related topics: Education, Computers

computer science [uncountable]

the study of computers and what they can do:

a BSc in Computer Science

Figure 2.1: An entry in the Longman Dictionary of Contemporary English

2.1.2 Thesauri

A thesaurus is a reference work that lists words grouped together according to similarity of meanings. The notion of a thesaurus was conceived by Dr. Peter Mark Roget, who described it as being the converse of a dictionary. A dictionary explains the meaning of words, whereas a thesaurus aids in finding the words that best express an idea or meaning. Some related work uses published thesauri such as the Roget's Thesaurus and the Macquarie Thesaurus for computation of semantics.

2.1.2.1 Roget's Thesaurus

Roget's Thesaurus is a widely-used English language thesaurus. It was created by Dr. Roget in 1805 and released to the public on April 29, 1852. The original edition had 15,000 words. The Karpeles Manuscript Library Museum[1] houses the original manuscript (Figure 2.2) in its collection. An electronic version of the Roget's Thesaurus is offered by the Project Gutenberg.[2] *Roget's Thesaurus* has a hierarchical structure that starts with a few major classes. Each class is further divided into subclasses. The 1911 edition of *Roget's Thesaurus of English Words and Phrases* is composed of six primary classes: (1) abstract relations, (2) space, (3) matter, (4) intellectual faculties, (5) voluntary powers, and (6) sentiment and moral powers.[3]

---

[1] `http://www.rain.org/~karpeles/rogetdis.html`

[2] `http://www.gutenberg.org/ebooks/10681`

[3] `http://poets.notredame.ac.jp/Roget/contents.html`

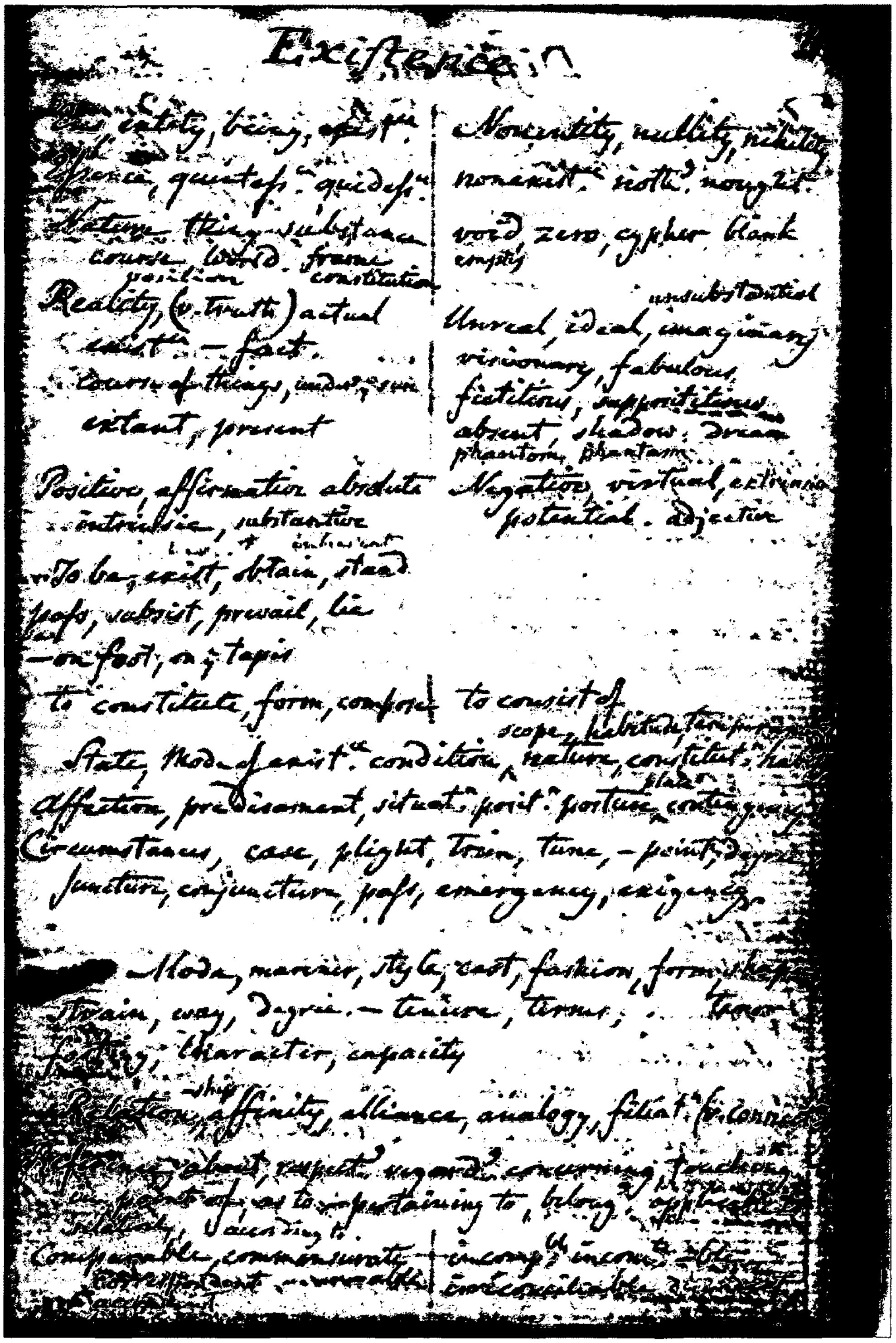

Existence
Ens, entity, being, existence | Nonentity, nullity, nihility
essence, quintessence, quiddity | nonexistence, nothing, nought
Nature, thing, substance | void, zero, cypher, blank
course, world, frame, position, constitution | empty
Reality, (v. truth) actual | Unreal, ideal, imaginary, unsubstantial
existence — fact | visionary, fabulous
course of things | fictitious, supposititious
extant, present | absent, shadow, dream, phantom, phantasm
Positive, affirmative, absolute | Negative, virtual, extrinsic
intrinsic, substantive | potential, adjective
To be, exist, obtain, stand
pass, subsist, prevail, lie
— on foot, on the tapis
to constitute, form, compose | to consist of
State, mode of existence, condition, nature, constitution, scope
affection, predicament, situation, position, posture, place, contingency
Circumstances, case, plight, trim, tune, — point, degree
Juncture, conjuncture, pass, emergency, exigency
Mode, manner, style, cast, fashion, form, shape
strain, way, degree — tenure, terms
character, capacity
Relation, affinity, alliance, analogy, filiation, connection
Reference, about, respecting, regarding, concerning, touching
in point of, as to, appertaining to, belonging, applicable
according to
Comparable, commensurate | incomparable, incommensurable
correspondent | irreconcilable

Figure 2.2: Original manuscript of the Roget's Thesaurus from the Karpeles Manuscript Library Museum

Each class is composed of multiple divisions and then sections, with a total of 1043 entries. Each entry maintains a group of words with similar meanings.

2.1.2.2 Macquarie Thesaurus

The *Macquarie Thesaurus* [2] is the first thesaurus written to be based on the distinctly Australian use of English (Figure 2.3). The full edition of the Macquarie Thesaurus consists of over 800 keywords and over 200,000 *synonyms* in English.[4] Besides hard copies, the Macquarie Thesaurus is available in ASCII, SGML and XML formats.

2.1.3 WordNet

WordNet [18] is a publicly available English lexical database which groups nouns, verbs, adjectives, and adverbs into sets of cognitive *synonyms* (*synsets*), each expressing a distinct concept. Started developing in 1985, WordNet is often regarded as an *ontology* [24] for natural languages. WordNet 3.0 contains a total of 117,659 synsets that are mostly nouns (82,115 nouns, 13,767 verbs, 18,156 adjectives, and 3,621 adverbs). Synsets are interlinked through semantic and lexical relations. The main relation among synsets in WordNet is *synonymy*. Other relations include *hyponymy*, *hypernymy*, *holonymy*, *meronymy*, and *antonymy*.[5] The WordNet taxonomy can be regarded as a tree, where the root node is the "entity" synset. The deeper a synset's position in the tree, the more specific it is. Users can download a copy of WordNet and run it locally, or manually access it on-line[6] (Figure 2.4). WordNet is manually maintained by the Global WordNet Association[7] and is available in different natural

---

[4] `http://www.macquariedictionary.com.au/`

[5] The definitions of hyponymy, hypernymy, holonymy, meronymy, and antonymy are given in the Glossary (page 104).

[6] `http://wordnetweb.princeton.edu/perl/webwn`

[7] `http://www.globalwordnet.org/`

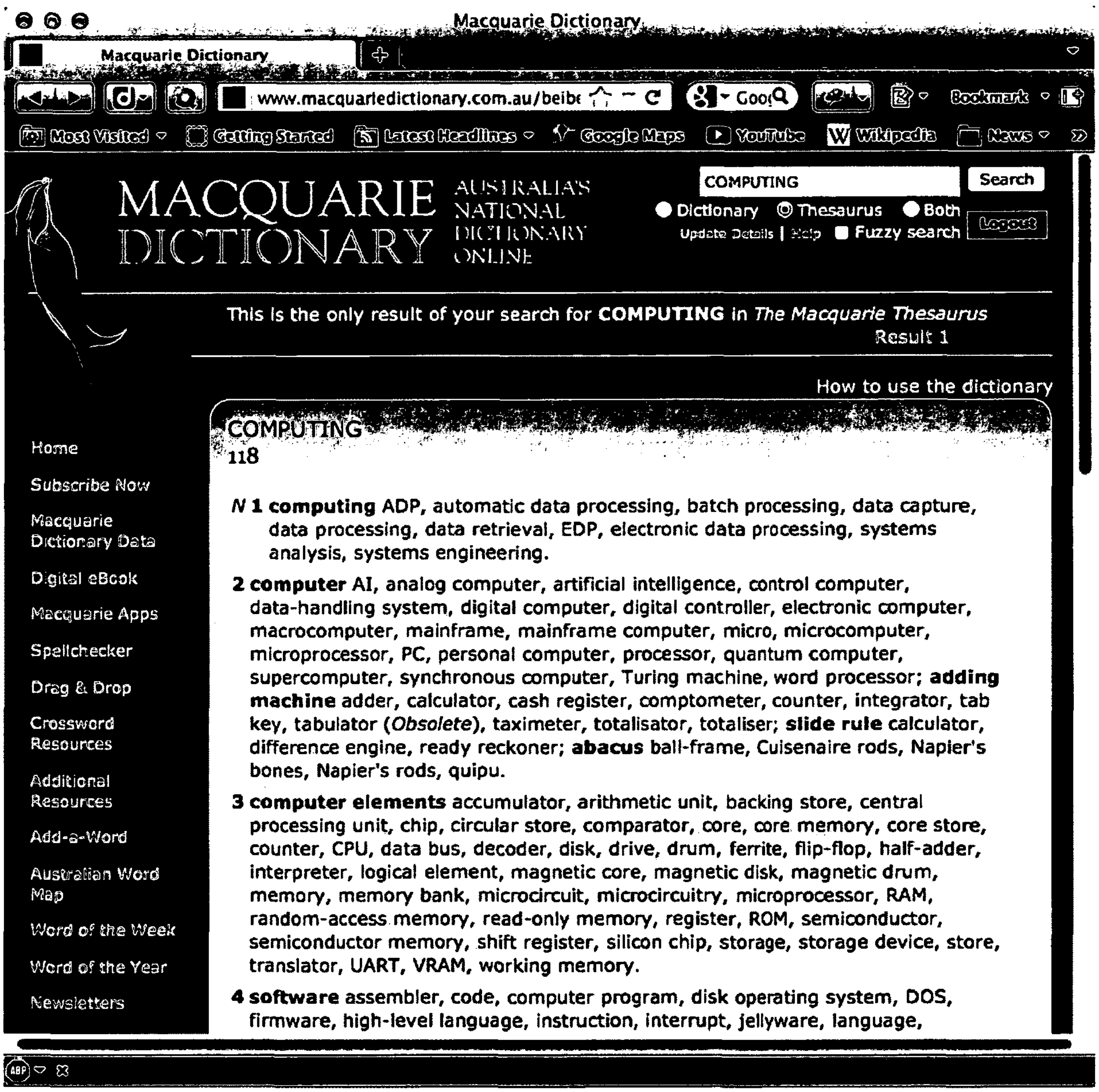


Figure 2.3: On-line version of the Macquarie Thesaurus

languages. Its API is available in over 20 programming languages and environments.[8]

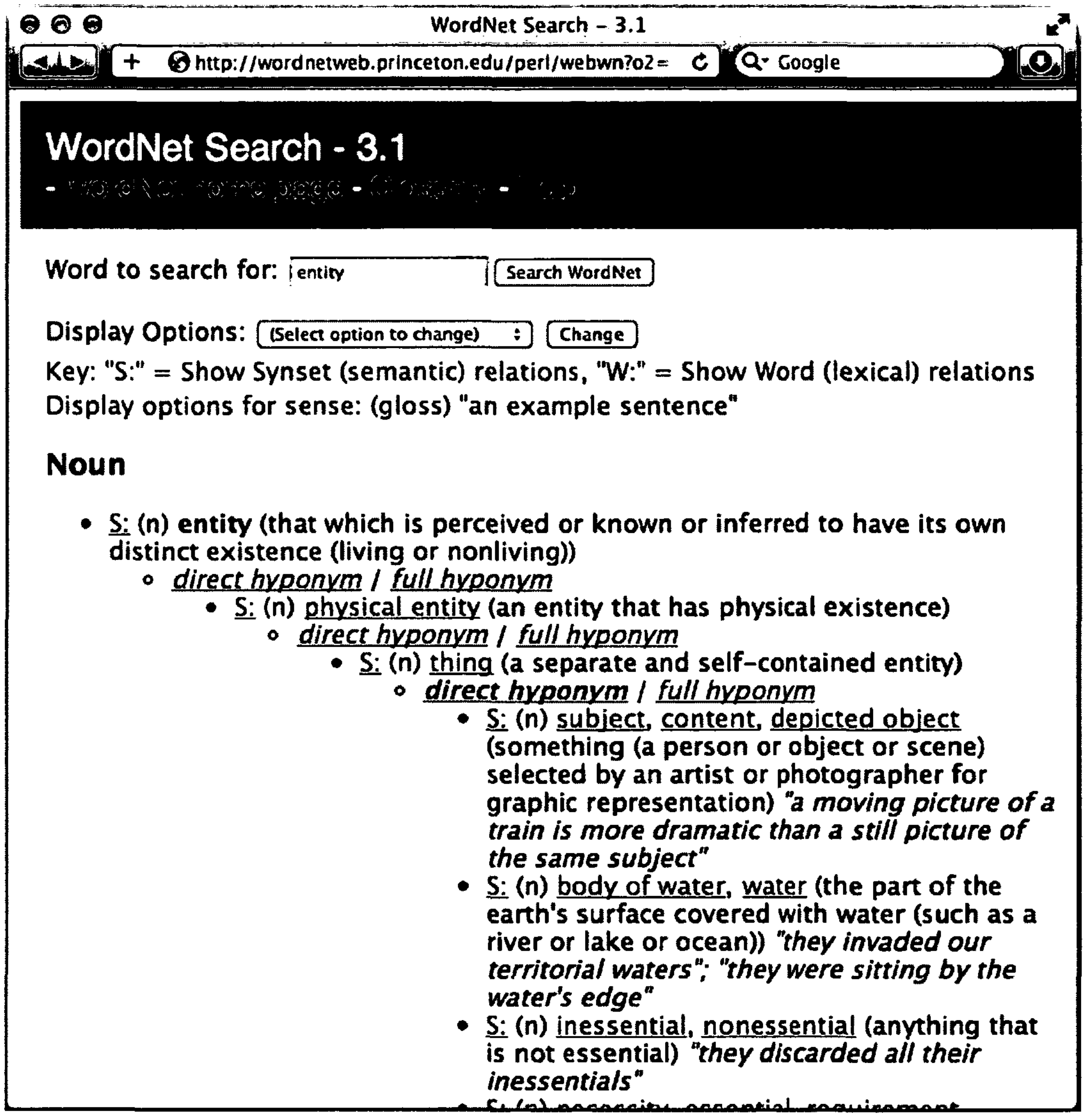


Figure 2.4: WordNet's on-line version

2.1.4 Cyc

Started in 1984, *Cyc* [34] is a multi-contextual knowledge base and inference engine developed by Cycorp.[9] It attempts to assemble a comprehensive *ontology* of every-

[8] `http://wordnet.princeton.edu/wordnet/related-projects/`

[9] `http://www.cyc.com/cyc/technology/whatiscyc`

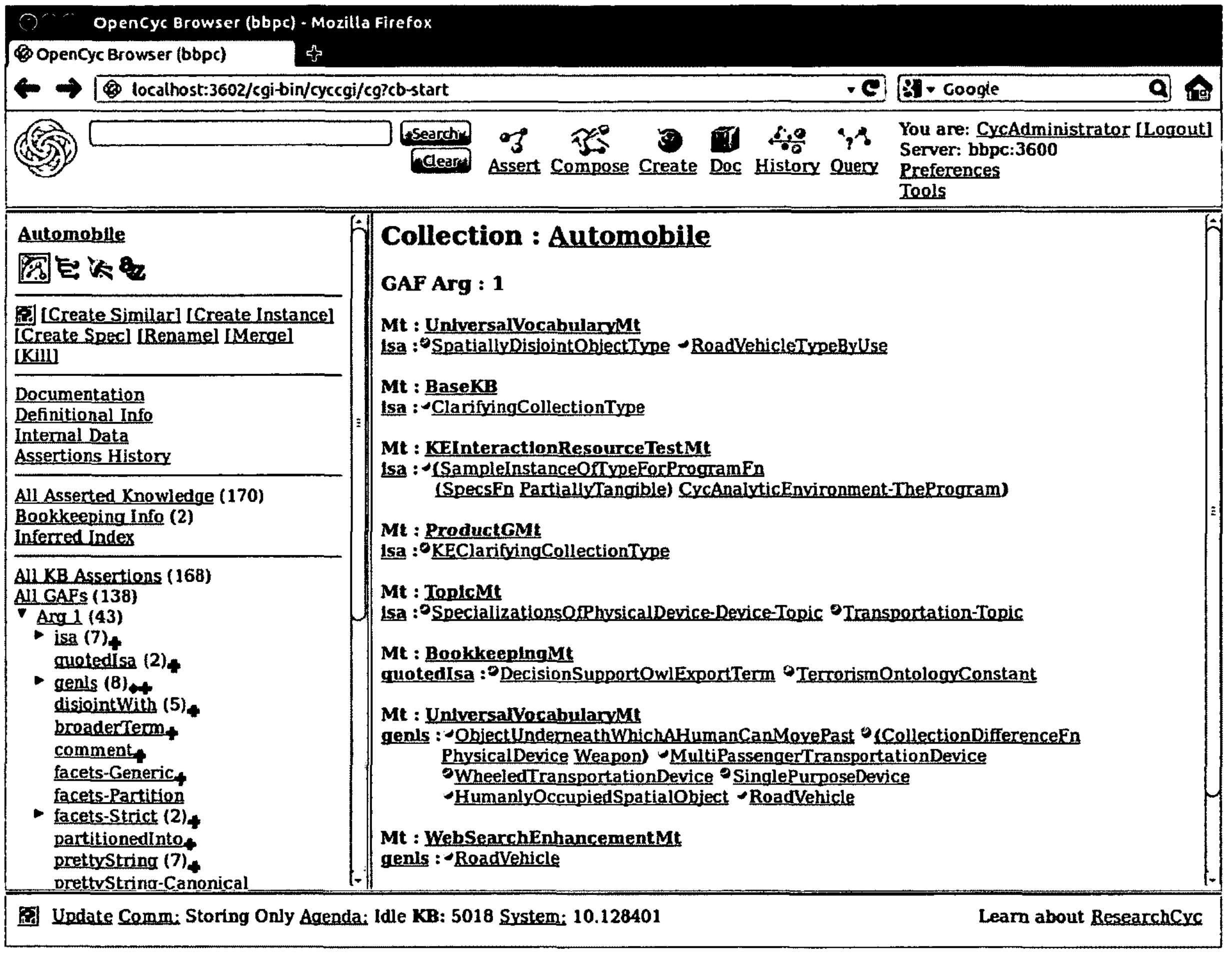


Figure 2.5: OpenCyc knowledge base browser

day common sense knowledge, with the goal of enabling AI applications to perform human-like reasoning. *OpenCyc*[10] is an open source version of the Cyc knowledge server, which is a unilingual form based on English that includes an inference engine, a knowledge base browser (Figure 2.5), and an API for writing programs in high-level languages that access and use the knowledge server.

Similar to WordNet, Cyc (including OpenCyc) is manually maintained. OpenCyc contains primarily definitional assertions that position concepts in the ontology and semantically constrain their use within assertions, alternate expressions of each term, and links between its concepts and those in selected semantic web ontologies.

A Cyc application is typically made up of several parts: the base of facts and rules, a set of queries (which could be complete queries or partial queries called *query templates*), and an external program written in a high-level language that interacts with the Cyc knowledge base and the user.

2.2 Corpus-based Resources

A *corpus* (plural *corpora*) in linguistics is a large and structured set of texts. Many corpora are designed to balance materials from one or more genres. They are widely used in computational linguistics for pattern learning and hypothesis testing. Some corpora are collections of raw text files while the others are annotated with syntactic structures. An annotated corpus is sometimes called a *parsed corpus*, or a *treebank*. Below are some popular corpora used for semantic relatedness measurement in related work.

2.2.1 Project Gutenberg

Project Gutenberg (`http://www.gutenberg.org/`) is the oldest and largest project to make copyrighted literature freely available online. Project Gutenberg digitized

[10] `http://www.opencyc.org/`

and proofread books with the help of thousands of volunteers. The catalog contains over 38,000 free books on a wide range of topics. Most of the books are available in formats such as HTML, EPUB, KINDLE, and plain text (Figure 2.6). Users can choose to download e-books or read them online.

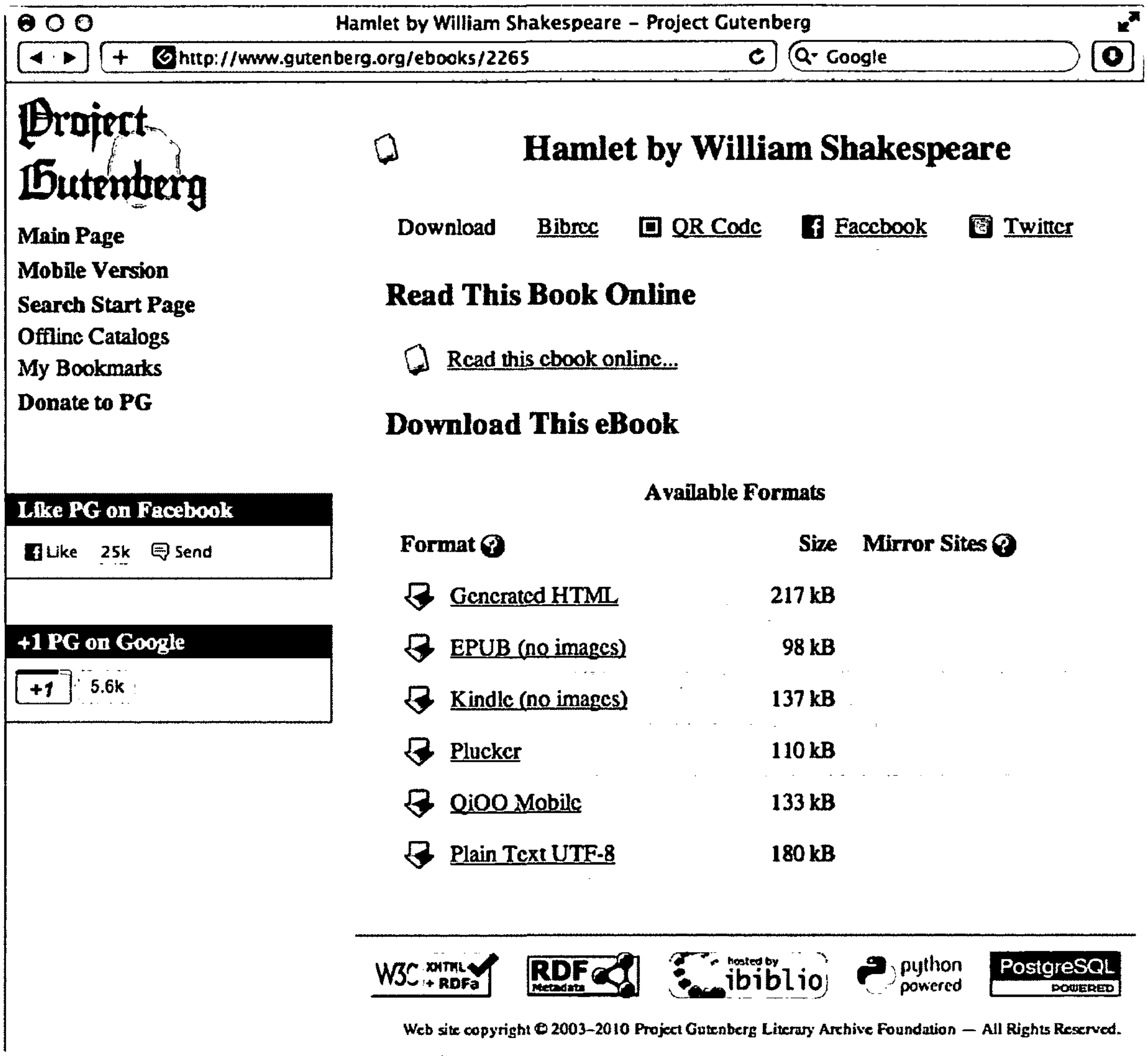


Figure 2.6: Project Gutenberg publishes e-books in various formats

2.2.2 British National Corpus

The British National Corpus (BNC) is a 100 million word collection of samples of written and spoken language from a wide range of sources in modern British En-

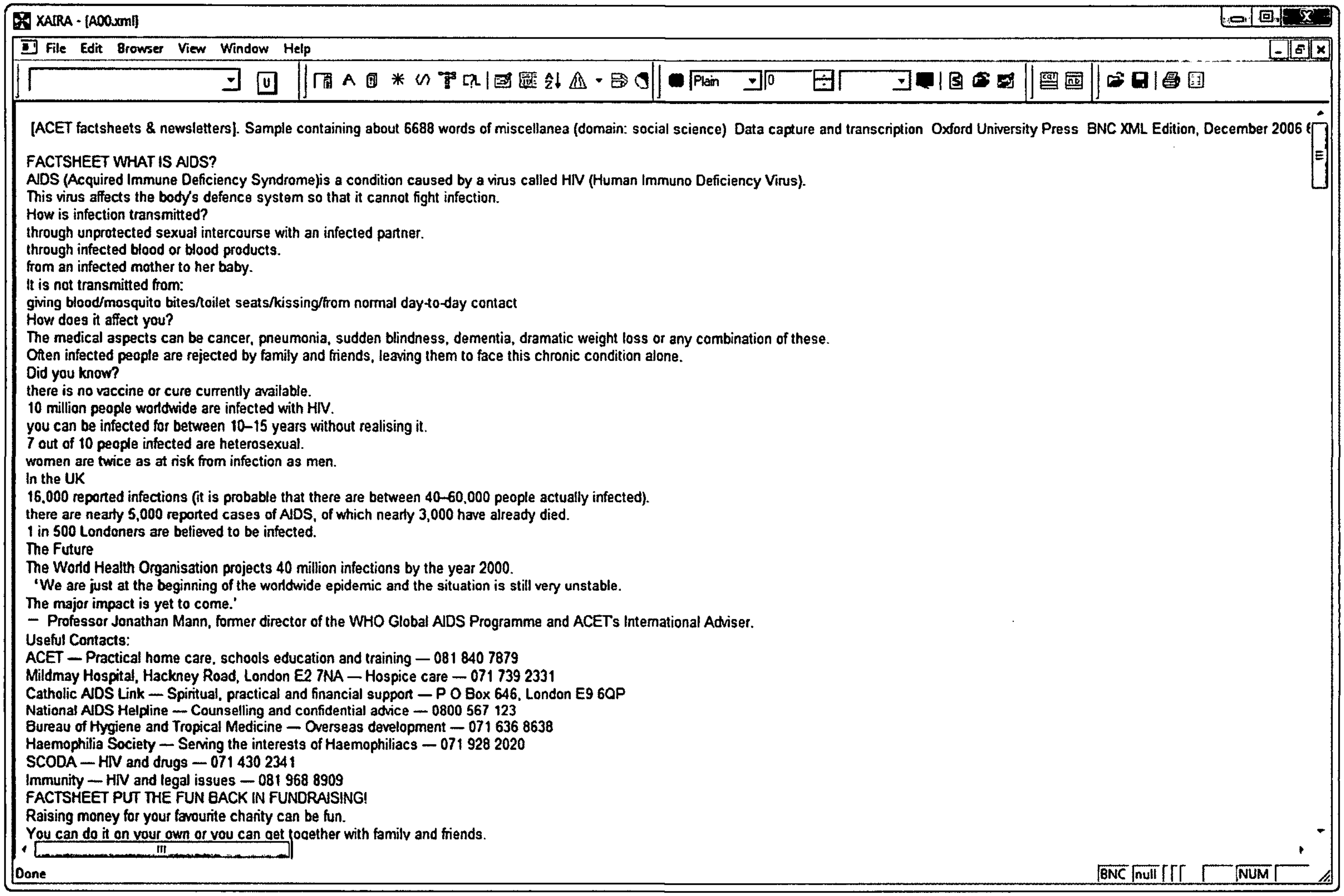


Figure 2.7: A fragment of the BNC XML edition rendered by the Xaira reader

glish [11]. The written part dominates 90% of the corpus, and it includes sources such as newspapers, journals, academic books, school and university essays, among many other kinds of text. The spoken part contributes 10% of the corpus. It consists of transcriptions of informal conversations and spoken language collected in different contexts from formal business or government meetings to radio shows and phone-ins. A copy of the BNC can be obtained from `http://www.natcorp.ox.ac.uk` and comes with the BNC XML corpus and the Xaira XML reader (Figure 2.7).

2.2.3 Penn Treebank

The *Penn Treebank project* (`http://www.cis.upenn.edu/~treebank/`) is the first large-scale treebank. It annotates corpora (such as the Wall Street Journal, the Brown Corpus [20], Switchboard, and ATIS) for linguistic structures. The annotated text can be searched with the tgrep[11] program by accessing LDC Online.[12] The Penn Treebank *part of speech* (POS) tags (Appendix A) are commonly adopted for NLP tasks involving POS tagging.

2.3 Hybrid Resources

Other resources such as Wikipedia and Wiktionary represent both lexicographical resources and corpora. These are therefore categorized as hybrid resources.

2.3.1 Wikipedia

Wikipedia (`http://www.wikipedia.org`), previously known as Nupedia, is a web-based, multilingual encyclopedia project based on an openly editable model. Anyone with Internet access can write and make changes to most Wikipedia articles. Users can contribute anonymously, or under a pseudonym, or with their real iden-

[11] `http://www.ldc.upenn.edu/ldc/online/treebank/`

[12] LDC Online: `https://online.ldc.upenn.edu/`

tity. Wikipedia was launched in January 2001 by Jimmy Wales and Larry Sanger. Since then, Wikipedia has grown rapidly (Figure 2.8) into one of the largest reference websites. As of March 5, 2012, Wikipedia was available in 284 languages, and the English Wikipedia contained 3,889,373 articles. Wikipedia is operated by the non-profit Wikimedia Foundation (`http://wikimediafoundation.org/`).

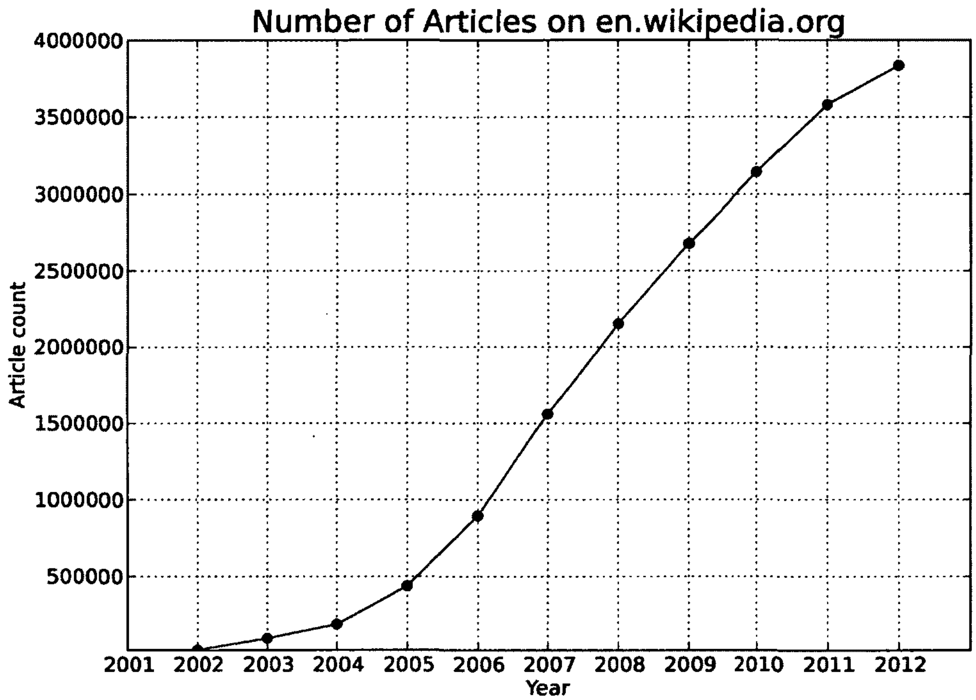


Figure 2.8: The number of articles on en.wikipedia.org grows exponentially

Wikipedia is based on the MediaWiki, a free open source wiki package used by several projects of the Wikimedia Foundation and by many other wikis. A copy of the MediaWiki package can be obtained from `http://www.mediawiki.org/`.

In recent years, there has been increasing interest in applying Wikipedia and related resources to question answering [12], word sense disambiguation (WSD) [43], named entity disambiguation [51], *ontology* evaluation [70], semantic web [65], and computing semantic relatedness [53]. In the field of semantic relatedness measurement, related work has used Wikipedia as a corpus by going through the content of each page, or as a lexicographical resource by parsing the category taxonomy. Ponzetto and Strube [53] deduce semantic relatedness of words by modeling relations on the Wikipedia category graph (Section 3.1.3.6, page 42). Gabrilovich and Markovitch [22] introduce the Explicit Semantic Analysis (ESA) model which calculates *TF-IDF*[13] values [58, 40] for every word and every document in Wikipedia and further uses local linkage information to build a second-level semantic interpreter (Section 3.1.2.5, page 35). These approaches are found to perform better than previous work based on traditional knowledge sources.

Because Wikipedia is openly-editable by anyone anywhere, the *knowledge acquisition bottleneck* (Section 1.2, page 3) existing in traditional knowledge sources becomes a minor problem for Wikipedia. Some related work even boldly states that Wikipedia solves the knowledge acquisition bottleneck [53]. Truly, Wikipedia's openly-editable model keeps its content up-to-date with the human knowledge so that the coverage problem (Section 1.2) in the knowledge acquisition bottleneck is no longer an issue. However, with the continuous page edits that are sometimes correct and sometimes unintentionally or intentionally wrong, and with the continuous growth of most pages in length, the difficulties of knowledge acquisition, and more specifically knowledge inaccuracy and maintenance trap (Section 1.2), still exist in Wikipedia.

2.3.1.1 Anatomy of a Wikipedia Article

Figure 2.9 shows a browser view of a Wikipedia page. A Wikipedia page contains its page title and page content. The title can be a unique identifier of the page. The content contains the detailed description of the page title. It also includes hyperlinks to other Wikipedia pages. These links are called *outlinks*. A page usually belongs to one or more *categories*. These categories are listed near the bottom of

[13]TF-IDF stands for Term Frequency–Inverse Document Frequency, a weighting scheme often used in information retrieval and text mining.

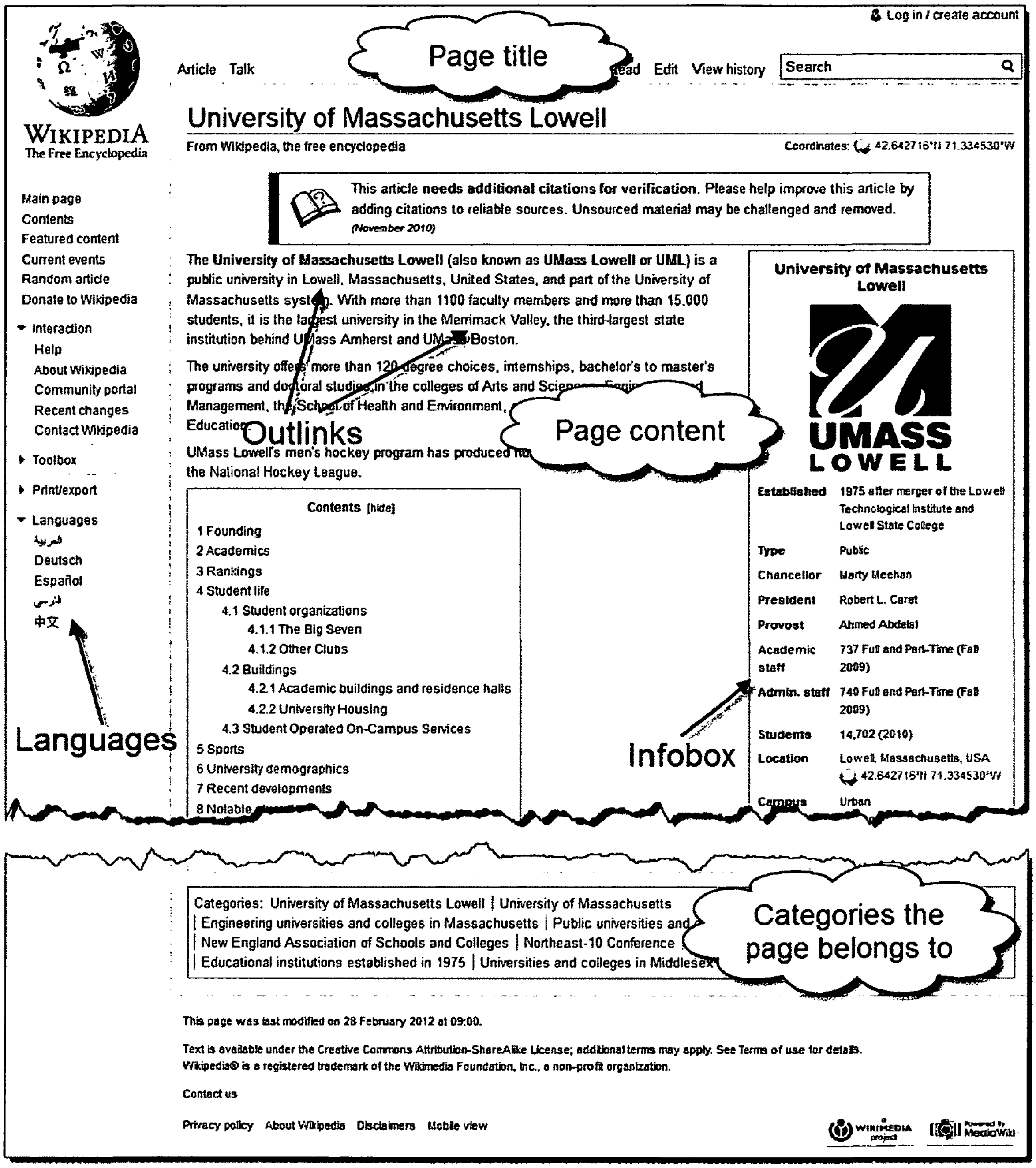


Figure 2.9: Anatomy of a Wikipedia Article

the page. Pages are grouped into categories by their conceptual relatedness. For example, page "University of Massachusetts Lowell" belongs to categories such as "Category:University of Massachusetts" and "Category:Universities and colleges in Middlesex County, Massachusetts."

A page may be available in another language by clicking the corresponding language on the left of the page.

The content of a page may optionally have an *infobox*, which is a tabular summary of the object's key attributes [65]. For example, the infobox for the "University of Massachusetts Lowell" page contains attributes such as the university logo, year of establishment, chancellor name, and the geographical location.

2.3.2 Wiktionary

Wiktionary (`http://www.wiktionary.org`) is a sister project of Wikipedia that is run by the Wikimedia Foundation. It is a multilingual, web-based project to create a free content dictionary. The structure of a Wiktionary page is very similar to that of Wikipedia, in that a page includes its page title, description, and the categories this page falls into. Wiktionary was brought online on December 12, 2002, following a proposal by Daniel Alston and an idea by Larry Sanger, co-founder of Wikipedia. So far, Wiktionary is available in 158 languages. The largest is the English Wiktionary, with over 2.5 million entries. Zesch et al. [71] show that Wiktionary is the best lexical semantic resource in the ranking task and performs comparably to other resources (such as WordNet and Wikipedia) in the word choice task.

CHAPTER 3

RELATED WORK

3.1 Semantic Relatedness

Semantic relatedness has been used in applications such as word sense disambiguation, named entity disambiguation, text summarization and annotation, lexical selection, automatic spelling correction, and text structure evaluation. These applications represent different strategies designed to evolve the current web into a semantic web, i.e., to turn existing web resources into knowledge-based structures. A semantic relatedness measure is a mapping $\varphi : w_1, w_2 \rightarrow n, n \in [0, d]$, where the inputs w_1 and w_2 are two terms, and the output n is a normalized metric value between 0.0 and d (d is typically 1). Output $n = d$ if the two terms are synonyms, and $n = 0$ if they are semantically unrelated.

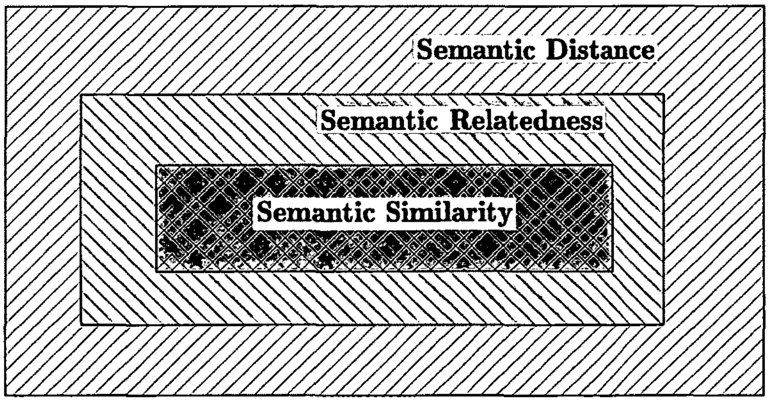


Figure 3.1: The relations of semantic distance, semantic relatedness, and semantic similarity as described by Budanitsky and Hirst [9].

Three terms are used interchangeably in related literature: semantic relatedness, semantic similarity, and semantic distance. Their relations are shown in Figure 3.1. Semantic relatedness is more generic than semantic similarity in that it includes all classical and non-classical semantic relations such as *holonymy*, *meronymy*, and *antonymy*, while semantic similarity is limited to relations such as *hyponymy* and *hypernymy*. Although an inverse of semantic relatedness, semantic distance has been used in related work on either just similarity or relatedness in general [9]. This study focuses on the closeness of concepts and considers both hyponymy/hypernymy and holonymy/meronymy relations. Therefore, the term semantic relatedness is applied to this study.

Given a taxonomy expressed as an *IS_A network*,[1] a straightforward method to calculate the relatedness between two words or phrases is to build a function based on the length of the shortest path from one node to the other [7, 55]. That is, the shorter the path from one node to another in the taxonomy, the more related they are. This method is formally known as an edge counting method, and it can be traced back to the semantic memory model proposed by Collins and Quillian [15] in 1969. Rada et al. [55] show that shortest path lengths measure conceptual distance better on IS_A links than on Quillian's model of semantic memory. They also prove that the minimum number of edges between two concepts is a metric for measuring their conceptual distance. Their work forms the basis of edge counting-based relatedness methods. Generally the path distance relatedness of two words in a taxonomy is defined as:

$$S(w_1, w_2) = \frac{1}{Dist(w_1, w_2) + 1}, \tag{3.1}$$

[1]IS_A networks are broadly used in areas such as artificial intelligence, databases, and software engineering for knowledge representation and software design. If concept A is a logical subclass of concept B, we say that A and B have an IS_A link. An IS_A network is a hierarchical structure of these IS_A links.

where w_1 and w_2 are two words, and $Dist(w_1, w_2)$ is the shortest distance between w_1 and w_2.

For example, in the taxonomy shown in Figure 3.2, $Dist(cat, fish) = 3$, $Sim(cat, fish) = 1/(3+1) = 0.25$. Similarly, $Sim(cat, apple) = 1/(8+1) = 0.11$. Since $Sim(cat, fish) > Sim(cat, apple)$, "cat" is semantically more related to "fish" than "apple."

However, this edge counting method makes a naïve assumption that words or *concept nodes* are uniformly distributed, which is not realistic in some scenarios. Other work suggests relatedness using metrics such as *information content* and co-occurrence. This study divides related research into three categories (as shown in Table 3.1).

Type of Methods	**Section**
Lexicographic resources only	Section 3.1.1
Corpora only	Section 3.1.2
Both lexicographic resources and corpora	Section 3.1.3

Table 3.1: Types of semantic relatedness measures

The rest of this chapter shows some related work from the three categories. Note that if a method treats text as an unordered collection of words and ignores other information such as word orders and grammars, it follows a model that is formally known as the *bag-of-words model.*

3.1.1 Methods Based Solely On Lexicographic Resources

A semantic relatedness measure based on lexical information typically constructs a tree or an undirected or directed graph as the resource (i.e., a lexicographic resource), and computes relatedness on the properties of that resource. According to the comprehensive survey by Budanitsky and Hirst [9], three types of lexicographic resources are used in previous work to measure semantic relatedness: (1) dictionaries, (2) thesauri, and (3) semantic networks such as WordNet.

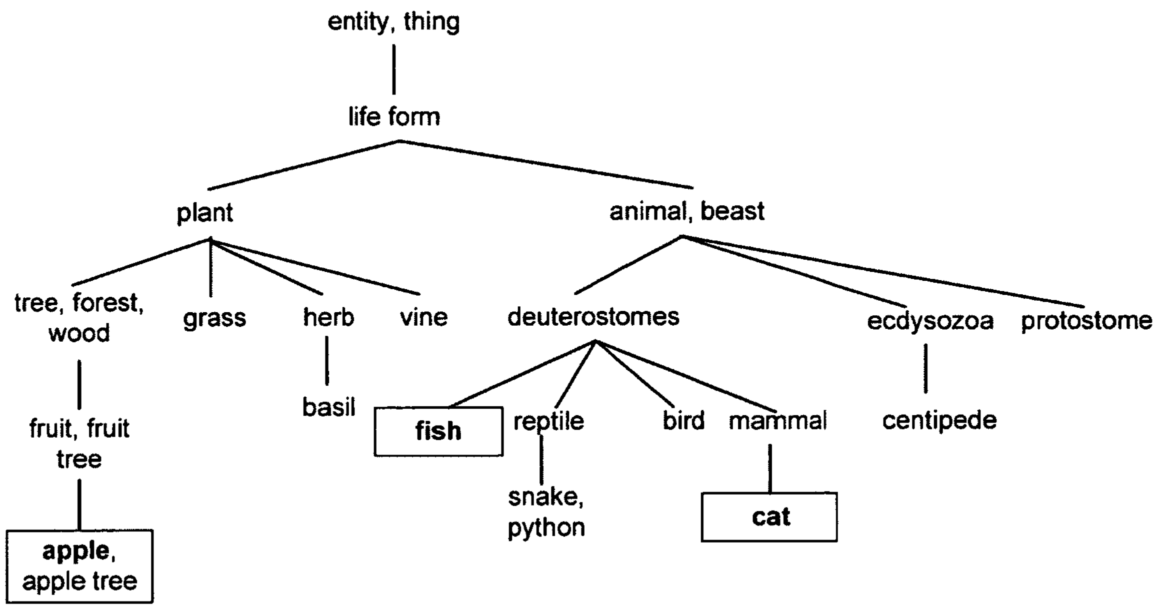


Figure 3.2: An IS_A hierarchical semantic knowledge base.

3.1.1.1 Dictionary-based

Kozima and Furugori [31] measure semantic similarity based on a semantic network of 2,851 nodes and 295,914 links constructed from the Longman Dictionary of Contemporary English (LDOCE, Section 2.1.1, page 7). The semantic network is constructed by creating a node for every word and linking each node to the nodes for all the words used in its definition. Similarity between words in the defining vocabulary is computed by means of spreading activation on this network. The semantic function is defined as a product of normalized frequency information and activity values. Since a dictionary does not explicitly provide categories that each word belongs to, a semantic relatedness measure based on a dictionary generally deduces how related two words are by analyzing the relatedness of their definitions in the dictionary, which may be misleading or too short to compare properly.

3.1.1.2 Thesaurus-based

Thesauri such as Roget's Thesaurus (Section 2.1.2.1) and the Macquarie Thesaurus (Section 2.1.2.2) group words into broad, loosely defined classes based on categories within which there are several levels of finer clustering. Although the classes and categories are named, the finer divisions are not. The words are clustered without attempting to explicitly indicate how and why they are related. For example, Figure 2.3 (Section 2.1.2.2, page 11) shows that the Macquarie Thesaurus groups terms "artificial intelligence" and "word processor" into the class "computer," but the fact that the two terms are related to "computer" differently is not distinguished by the thesaurus. Morris and Hirst [48] point out that related words might not be physically close in a thesaurus, and although physical closeness is important, "words in the index of the thesaurus often have widely scattered categories, and each category often points to a widely scattered selection of categories." Thesauri do not have to name or classify the relationship of words in the same category. A thesaurus-based semantic relatedness method using category structures and cross-references typically returns boolean values (such as "close" or "not close") instead of the traditional numeric value between 0 and 1.

3.1.1.3 WordNet-based

WordNet (Section 2.1.3) is commonly used as a lexicographic resource to calculate semantic relatedness. Figure 3.3 shows a fragment of the WordNet taxonomy. Only one *sense* of each *polysemous* word is displayed. A solid line indicates the hypernym-hyponym relation, while a dotted line indicates the holonym-meronym relation.[2] Each concept is formatted as $x.y.z$, where x is a word, y is either a noun (n) or verb (v), and z corresponds to a sense of word x. A WordNet-based method uses one or more edge-counting techniques in the WordNet taxonomy. The relatedness of two concept

[2]Word "gem" is a part (meronym) of "jewelry."

nodes is a function of the minimum number of hops between them.

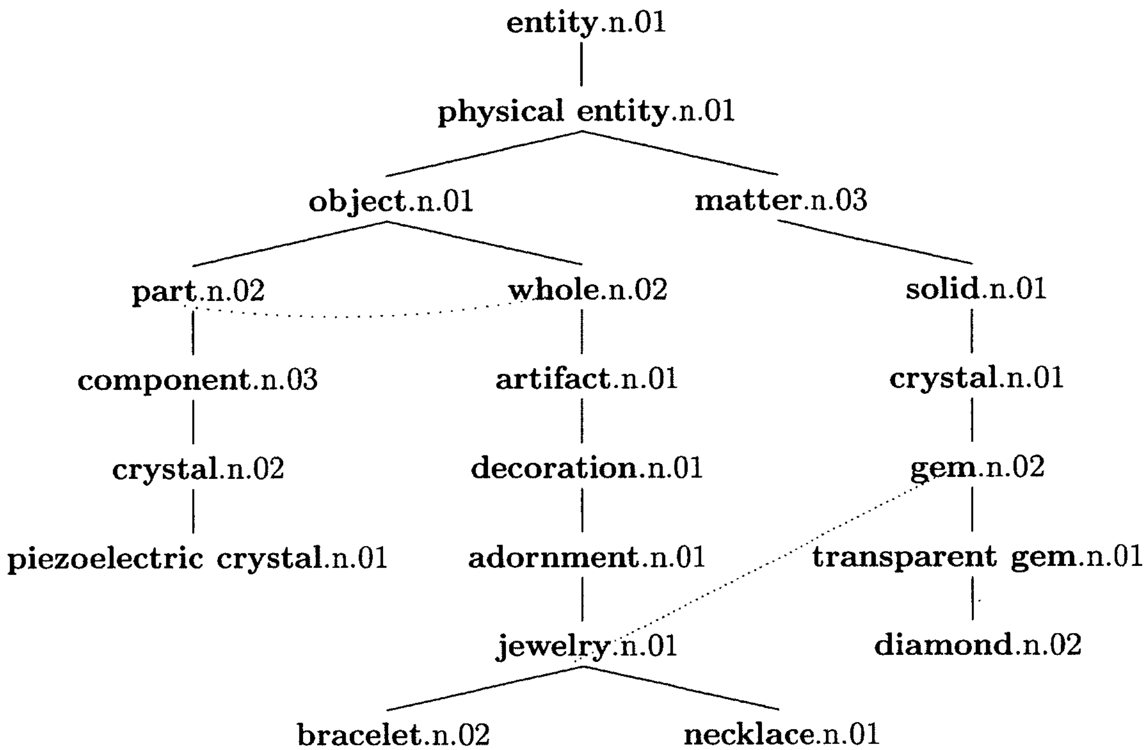


Figure 3.3: A fragment of the WordNet taxonomy.

3.1.1.3.1 Wu and Palmer's Conceptual Similarity Model

Wu and Palmer [66] address the problem of translating English verbs into Mandarin Chinese by using what they call conceptual similarity between a pair of concepts in the projected domain hierarchy. The conceptual similarity of two concepts is defined as:

$$S_{WP}(c_1, c_2) = \frac{2 * Dep(Lca(c_1, c_2))}{Dep(c_1) + Dep(c_2)}, \tag{3.2}$$

where c_1 and c_2 are two concept nodes in the hierarchy, $Lca(c_1, c_2)$ is the *lowest common ancestor* (LCA) of c_1 and c_2, and Dep is the depth of a concept node relative to the root. Note that the LCA does not necessarily appear in the shortest path connecting the two concept nodes, as it is by definition the common ancestor deepest in the taxonomy, not closest to the two concepts.

3.1.1.3.2 Leacock and Chodorow's Model

Leacock and Chodorow [33] propose a model based on the shortest path that connects concept nodes and the maximum depth of the taxonomy in which the concept nodes occur. They use the following model to compute the semantic similarity between concepts c_1 and c_2 in WordNet:

$$S_{LC}(c_1, c_2) = -\log \frac{Dist(c_1, c_2)}{2 * D}, \tag{3.3}$$

where $Dist(c_1, c_2)$ is the shortest distance between c_1 and c_2, D is a constant representing the maximum depth in the WordNet hierarchy, and $S_{LC}(c_1, c_2) \in [0, +\infty)$. Output $S_{LC}(c_1, c_2)$ is larger when c_1 and c_2 have a shorter distance, and $S_{LC}(c_1, c_2) = 0$ if the distance between c_1 and c_2 is twice the depth of the WordNet hierarchy.

3.1.1.3.3 Hirst and St-Onge's Lexical Chain Model

Hirst and St-Onge [27] propose a semantic relatedness model based on lexical chains of WordNet for the detection and correction of *malapropisms*. They distinguish three kinds of strengths of semantic relations in WordNet: extra-strong, strong, and medium-strong. An *extra-strong* relation holds "only between a word and its literal repetition." Two words have a *strong* relation if one of the following applies:

1. The two words have at least one synset in common.
2. Synsets of the two words are connected by the *antonymy* relation.
3. One of the two words contains the other.

A *medium-strong* relation between two words occurs when there exists a valid path connecting a synset associated with one word to another synset associated with the other word, and the valid path contains no more than five links and conforms to one of the eight patterns.

Words that are extra-strong or strong have uniform weights. On the other hand, words that are medium-strong are assigned different weights by the following formula:

$$Weight_{HS}(c_1, c_2) = C - Dist(c_1, c_2) - k * turns(c_1, c_2), \tag{3.4}$$

where C and k are two constants, and $turns(c_1, c_2)$ is the number of times the path between the two words changes its direction. Therefore, two words are assigned a lower weight if they have a longer path and more changes of direction over the WordNet taxonomy.

3.1.1.3.4 Yang and Powers's Model

Yang and Powers [69] propose a semantic relatedness model based on edge-counting that takes into account the part and whole (i.e., meronymy and holonymy) relations. Their model includes two searching algorithms over the WordNet taxonomy: bidirectional depth-limited search and uni-directional breadth-first search. They define the similarity of two concepts as:

$$Sim(c_1, c_2) = \begin{cases} \alpha_t \prod_{i=1}^{Dist(c_1,c_2)} \beta_{t_i} & \text{if } Dist(c_1, c_2) < \gamma \\ 0 & \text{if } Dist(c_1, c_2) \geq \gamma \end{cases} \tag{3.5}$$

where $Sim(c_1, c_2) \in [0, 1]$ and

- c_1, c_2: concept node 1 and concept node 2.
- $Dist(c_1, c_2)$: the shortest distance of c_1 and c_2.
- t: id (identity), hh (hypernym-hyponym), hm (holonym-meronym), or sa (synonym-antonym).
- α_t: a link type factor applied to a sequence of links of type t ($0 < \alpha_t \leq 1$).

- β_{t_i}: the path weight factor of a position i between c_1 and c_2, which also depends on the link type.

- γ: a user-defined threshold on the distance introduced for efficiency, representing human cognitive limitations.

Yang and Powers experimented with their approach against previous work by Resnik [56], Jiang and Conrath [29], and Lin [38]. Their results show that their proposed method performs the best over 28 pairs of nouns.

3.1.1.3.5 Seco et al.'s Information Content Model

Seco at al. [59] present a measure of information content that only relies on hierarchical structure in WordNet. They define the information content of a WordNet concept as a function of its hyponyms:

$$IC(c) = 1 - \frac{\log\left(hypo(c) + 1\right)}{\log\left(N\right)}, \tag{3.6}$$

where $hypo(c)$ is the number of hyponyms of a given concept c and N is the maximum number of concepts that exist in the taxonomy. Seco at al. state that their approach outperforms some of the previous work and one advantage of their approach is that "it does not rely on corpora analysis" therefore they "avoid the sparse data problem which is evident in many corpus based approaches" [59].

3.1.2 Methods Based Solely On Corpora

As explained in Section 2.2, a corpus is a large, structured set of texts collected from a wide range of sources, such as books, newspapers, web search engines, social networks, etc. Some related work is based on corpora collected from a search engine [57, 1, 13]. Other research uses corpora such as the British National Corpus (BNC) [11], Brown Corpus [20], and American National Corpus [28]. Such approaches are sometimes known as a subset of distributional measures [46]. One theory behind these approaches is the *distributional hypothesis* [19, 25]. The main point of this hypothesis is that there is a correlation between distributional similarity and meaning similarity. That is, two words are likely to be related if they co-occur within similar contexts.

This section reviews some of the popular corpus-based measures.

3.1.2.1 Query Expansion

Query expansion (QE) is a common way to measure semantic relatedness using web search engines. Given a seed query for input, QE expands the search query to match additional documents [8, 16]. The kernel function[3] developed by Sahami and Heilman [57] uses query expansion and accesses the Google corpus to generate additional suggestions for a given query. To calculate the QE for a query, their algorithm collects snippets[4] from a search engine and represents each snippet as a *TF-IDF* [58, 40] weighted term vector. The weight $w_{i,j}$ associated with term t_i in document d_j is defined by TF-IDF as:

$$w_{i,j} = tf_{i,j} \cdot \log(\frac{N}{df_i}), \tag{3.8}$$

where $tf_{i,j}$ is the frequency of t_i in d_j, N is the total number of documents in the corpus, and df_i is the total number of documents that contain t_i.

Each weighted term vector representing a snippet from the search engine is trun-

---

[3]A kernel function is defined as a function K such that for all $x, y \in X$

$$K(x, y) = \langle \phi(x) \cdot \phi(y) \rangle, \tag{3.7}$$

where ϕ is a mapping from X to an (inner product) feature space F [60]. A function is a kernel function if and only if it satisfies the Mercer's Theorem [60].

[4]A snippet is a small region of reusable text.

cated and L_2-normalized[5] to calculate the centroid. The $QE(s)$ of short text snippet s is the L_2 normalization of the centroid. They further define the semantic relatedness kernel function between two terms x and y as $K(x, y) = QE(x) \cdot QE(y)$.

Based on that kernel function, Abhishek and Hosanagar [1] have added keyword suggestions using an undirected semantic graph. Bollegala et al. [5] integrate both page counts and snippets to measure semantic similarity between word pairs.

Cilibrasi and Vitanyi propose the Normalized Google Distance (NGD) algorithm [13] to measure similarities of words and phrases from the WWW using Google page counts. Given two independent search terms x and y, their method makes queries to the Google search engine. Based on the page count N from Google, they define $f(x)$ to be the number of pages containing x, $f(y)$ to be the number of pages containing y, and $f(x, y)$ to be the number of pages containing both x and y. In turn, the NGD is defined by:

$$NGD(x, y) = \frac{\max(\log f(x), \log f(y)) - \log f(x, y)}{\log N - \min(\log f(x), \log f(y))} \tag{3.9}$$

The result of the NGD ranges from 0 to ∞.

3.1.2.2 LSA

Latent Semantic Analysis (LSA) [32] is a well-known corpus-based semantic similarity measure that is based on statistical information of words in a corpus. The underlying idea is that the aggregation of "all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other [32]." LSA represents text as a *word-by-context matrix* in which each row represents a unique

---

[5] The L_2-normalized form for a vector $X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ is defined as $\frac{X}{\|X\|_2} = \frac{X}{\sqrt{\sum_{i=1}^{n} |x_i|^2}}$.

word and each column represents a text passage. Each cell contains the frequency with which the word of its row appears in the passage denoted by the column. Next, the frequency of each cell is reweighed considering both the importance of the corresponding word in the text passage and the degree to which the word type carries information in the domain of discourse in general. The matrix is decomposed by the *singular value decomposition* (SVD) [23] (see Glossary on page 104) into the product of three new matrices: the first describes the original row entries as vectors of derived orthogonal factor values, the second describes the original column entries in the same way, and the third is a diagonal matrix containing scaling values. The dimensionality is reduced simply by deleting the smallest singular values in the diagonal matrix. The original word-by-context matrix is then reconstructed from the reduced dimensional space. Through the decomposition and reconstruction of the matrix, LSA acquires the context knowledge. To measure the similarity of two sentences, a vector for each sentence is formed in the reduced dimensional space, and the similarity is obtained using metrics such as the cosine coefficient between the two vectors.

Due to the computational limit of SVD, the dimension size of the LSA word-by-context matrix is limited to several hundred. Landauer et al. [32] state:

> LSA became practical only when computational power and algorithm efficiency improved sufficiently to support SVD of thousands of words-by-thousands of contexts matrices; it is still impossible to perform SVD on the hundreds of thousands by tens of millions matrices that would be needed to truly represent the sum of an adult's language exposure.

Despite the advances in computational power over recent years, LSA remains inefficient to execute especially over the *big data* [42].

LSA also has other drawbacks besides the inefficiency. LSA *only* induces its representations of the meaning of words and passages from analysis of input text. It does not use any manually constructed dictionaries, knowledge bases, semantic networks, grammars, syntactic parsers, or morphologies, or the like. By just focusing on the local text LSA ignores the big picture. Moreover, LSA represents the meaning of a word as the average of all its senses appearing in the text. The meaning of a text passage is regarded by LSA as the average meaning of all the words in it [32]. In other words, LSA cannot well capture *polysemous* words.

3.1.2.3 HAL

Similar to LSA, Hyperspace Analogues to Language (HAL) [10] is also based on statistical information of words in a corpus. HAL uses lexical co-occurrence information to construct a high-dimensional semantic space. In Burgess et al.'s work [10], a 10-word moving window is passed over a corpus of around 300 million words to record word co-occurrences. A word is assigned a higher weight if it is closer to the target word, and lower weight if it is farther away. HAL creates an $N \times N$ high-dimensional matrix where N is the number of unique words in the vocabulary. Each cell in the matrix stores the cumulative weight between a target word from the corresponding row and a word from the corresponding column. Next, a vector representing each word in $2N$ dimensions is formed by concatenating the transposition of a word's column with its row. A sentence vector is then created by adding the word vectors for all words in the sentence. Similarity between two sentences is calculated using a metric such as Euclidean distance. However, their experimental results show that HAL is not as promising as LSA on computation of similarity for short texts [10]. The construction of the high-dimensional memory matrix is expensive, and it may not capture a sentence's meaning well. Li et al. point out the drawback of HAL:

> HAL's drawback may be due to the building of the memory matrix and its approach to forming sentence vectors: The word-by-word matrix does not capture sentence meaning well and the sentence vector becomes diluted as a large number of words are added to it [37].

3.1.2.4 PMI-IR

Turney [61] proposes a Pointwise Mutual Information and Information Retrieval (PMI-IR) algorithm, an unsupervised learning algorithm for the identification of synonyms. Similar to LSA, PMI-IR is based on co-occurrences. The semantic relatedness of two words w_1 and w_2 by PMI-IR is defined as:

$$S_T(w_1, w_2) = \log_2 \frac{p(w_1, w_2)}{p(w_1)p(w_2)}, \tag{3.10}$$

where $p(w_1, w_2)$ is the probability that words w_1 and w_2 co-occur, and $p(w)$ is the probability of occurrence for word w. The probability of a word is calculated based on querying the word to the AltaVista search engine. For every synonym test question, Turney calculates $S_T(q, c)$ for the word in the question q and the word in each choice c. His work shows that PMI-IR receives a higher score than LSA in the evaluation of 130 synonym test questions collected from TOEFL and ESL exams.

3.1.2.5 ESA

Gabrilovich and Markovitch's Explicit Semantic Analysis (ESA) [21, 22] is a semantic relatedness measure built on top of Wikipedia (Section 2.3.1). Different from the latent concepts used by the LSA (Section 3.1.2.2), ESA explicitly uses the "knowledge collected and organized by humans" [22].

Given two text fragments as input, ESA constructs their 2-level semantic interpretation vectors and uses cosine coefficients to output the semantic relatedness score.

For a set of concepts $(C_1, C_2, \dots C_n)$ and their associated documents $(d_1, d_2, \dots d_n)$ in Wikipedia, the first level interpreter constructs a sparse table T where each column corresponds to a concept, each row corresponds to a word in all the documents, and an entry $T[i, j]$ in T corresponds to the TF-IDF value of term t_i in document d_j:

$$T[i, j] = tf(t_i, d_j) \cdot \log \frac{n}{df_i}, \tag{3.11}$$

where term frequency $tf(t_i, d_j)$ is a function of the number of times (*count*) t_i occurs in d_j:

$$tf(t_i, d_j) = \begin{cases} 1 + \log count(t_i, d_j) & \text{if } count(t_i, d_j) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

and df_i is the number of documents in the collection that contain the term t_i. The first level semantic interpreter of a text fragment is defined as the centroid of the vectors representing each word.

The second level interpreter takes into account the link structure in Wikipedia. A reduced weight is added to a term for each of its incoming links.

Because ESA tokenizes every word in every Wikipedia page to build the inverted document frequency for the first-level semantic interpreter, it is computationally expensive [68], especially considering the exponential growth of Wikipedia (Figure 2.8). In addition, although ESA's first-level semantic interpreter keeps the centroid of term vectors to perform partial *word sense disambiguation* (WSD), it neglects Wikipedia disambiguation, category, and redirection pages which contain semantic information that is useful for the refinement of WSD.

3.1.3 Hybrid Methods

Some related work combines lexicographic resources (such as WordNet) with corpus statistics [56, 29, 39]. It has been shown that these composite methods generally outperform lexicographic resource- and corpus-based methods [9, 17, 45]. They are classified as hybrid methods.

3.1.3.1 Resnik's Information Content Model

Resnik's model [56] is based on the idea that the similarity of two concepts in an IS_A taxonomy is the "extent to which they share information in common." Resnik's work points out that the edge counting method captures the shared information indirectly. If the minimal path of IS_A links between two nodes is long, that means it is necessary to go high in the taxonomy to more abstract concepts in order to find the lowest common ancestor. For example, if we just look at the hypernym-hyponym relation in Figure 3.3 (page 27), *bracelet* and *necklace* are both subsumed by *jewelry*, whereas the *lowest common ancestor* (LCA) of *bracelet* and *diamond* is *physical entity*. Two concepts are more similar if they have more information in common. The shared information of two concepts is indicated by the *information content* (IC) of their lowest common ancestor. Let $p(c)$ be the probability of encountering an instance of concept c, the IC of c is $-\log p(c)$. The semantic similarity of two concepts c_1 and c_2 is defined as:

$$S_R(c_1, c_2) = -\log p(Lca(c_1, c_2)), \tag{3.13}$$

where $Lca(c_1, c_2)$ is the *lowest common ancestor* of concepts c_1 and c_2.

Concept frequencies are estimated using noun frequencies from the Brown Corpus of American English. Each concept that occurs in the corpus is counted as an occurrence of itself as well as all of its ancestors. The probability $p(c)$ for a concept c is correlated to the concept frequency $freq(c)$:

$$p(c) = \frac{freq(c)}{N}, \tag{3.14}$$

where N is the total number of nouns observed excluding those not subsumed by any WordNet concepts.

Since the occurrence of a concept in the corpus not only increments its own frequency but also the frequencies of all its ancestors, the value of p increases as one moves up the WordNet taxonomy. That is, if concept c_1 IS_A c_2, then $p(c_1) \leq p(c_2)$. If the root is unique for a taxonomy, its probability will be 1. For example, since the WordNet taxonomy has a unique top node *entity.n.01*, its probability is 1. With Equation 3.13, two concepts sharing *entity.n.01* as the lowest common ancestor have a similarity of $-log(1) = 0$. The computation of information content is sometimes referred as the *node-based* approach (as opposed to the *edge-based* approach).

As Jiang and Conrath [29] point out, one shortcoming of Resnik's model is that it does not emphasize the importance of edges in the WordNet taxonomy. Edges are only used for locating the ancestors of a pair of concepts. Concepts sharing the same lowest common ancestor are not distinguished. In Figure 3.3 (page 27) for example, $S_R(part, whole) = S_R(part, bracelet)$ using Equation 3.13, because pairs (*part*, *whole*) and (*part*, *bracelet*) share the same lowest common ancestor *object*. This problem in turn decreases the accuracy especially when most of the concept nodes share the same LCA. In addition to the nodes, the number of edges in the taxonomy should also be considered.

3.1.3.2 Jiang and Conrath's Model

To address the problem in Resnik's information content model, Jiang and Conrath [29] combine the edge-based approach of the edge counting scheme with the node-based approach of the information content computation. Besides the WordNet taxonomy, their approach uses corpus statistics as a secondary source for corrections. Concept frequencies are estimated using noun frequencies from SemCor [44], a sense-tagged corpus built from a subset of the Brown Corpus.

They consider factors such as link type, depth, conceptual density, and information content of concepts to measure the semantic similarity. Edge weight for a concept node c and its parent p is defined as:

$$wt(c,p) = [\beta + (1-\beta) * \frac{\overline{E}}{E(p)}] * (1 + \frac{1}{dep(p)})^{\alpha}[IC(c) - IC(p)] * T(c,p), \tag{3.15}$$

where $dep(p)$ denotes the depth of node p in the WordNet hierarchy, $\overline{E}$ is the average density of the entire hierarchy, $E(p)$ is the number of edges from p, $IC(c)$ is the information content for node c, $T(c, p)$ is a link type factor, and parameters α ($\alpha \geq 0$) and β ($\beta \in [0, 1]$) control the degree to which the node depth and density factors contribute to the edge weighting scheme.

The overall semantic distance $S_{JC}(w_1, w_2)$ between two words w_1 and w_2 is the summation of edge weights along the shortest path between the concept nodes for the two words:

$$S_{JC}(w_1, w_2) = \sum_{c \in path(c_1, c_2) - Lca(c_1, c_2)} wt(c, parent(c)), \tag{3.16}$$

where concept c_1 is a sense of word w_1, c_2 is a sense of w_2, $path(c_1, c_2)$ is the set that contains all the nodes in the shortest path from c_1 to c_2, and $Lca(c_1, c_2)$ is the lowest common ancestor of c_1 and c_2.

If only edges are taken into account (i.e., $\alpha = 0$, $\beta = 1$, and $T(c, p) = 1$), the semantic distance is rewritten as:

$$S_{JC}(w_1, w_2) = IC(c_1) + IC(c_2) - 2 * IC(Lca(c_1, c_2)). \tag{3.17}$$

3.1.3.3 Lin's Model

Lin [39] points out that one drawback of previous semantic similarity measures is their dependency on a particular application or domain. He attempts to address the problem by proposing a model that is both universal (that can be applied to arbitrary domains and even those where "no similarity measure has previously been proposed") and theoretically justified (the measure is "not defined directly by a formula" and is instead "derived from a set of assumptions about similarity"). The model is based on three intuitions:

1. The similarity between A and B is related to their commonality. The more commonality they share, the more similar they are.

2. The similarity between A and B is related to the differences between them. The more differences they have, the less similar they are.

3. The maximum similarity between A and B is reached when A and B are identical, no matter how much commonality they share.

The similarity between two concept nodes in a taxonomy is determined by the ratio between the amount of information needed to state the commonality of the two nodes and the information needed to fully describe what they are. The similarity model can be simplified as:

$$S_{Lin}(c_1, c_2) = \frac{2 * IC(Lca(c_1, c_2))}{IC(c_1) + IC(c_2)}, \tag{3.18}$$

which is essentially a normalized form of the model by Jiang and Conrath (Section 3.1.3.2, page 38).

3.1.3.4 Mohammad and Hirst's Distributional Profiling Model

Mohammad and Hirst propose a hybrid approach [47, 45] that combines BNC (Section 2.2.2) corpus statistics with the Macquarie Thesaurus (Section 2.1.2.2) to calculate words' semantic distance. They argue that "estimating semantic distance is essentially a property of concepts (rather than words)" and that two concepts are semantically close if they share similar sets of words. Their argument is built on top of the *distributional hypothesis* which states that words that are semantically close tend to occur in similar contexts [19, 25].

To build the distributional profile of concepts for a keyword, they extract related words from BNC and corresponding categories from the Macquarie Thesaurus to construct a word-category co-occurrence matrix. Each row in the matrix corresponds to a word from the BNC, each column represents a category (or concept) from the thesaurus, and each entry of the matrix captures the number of times a category and a word co-occur. A new bootstrapped word-category co-occurrence matrix is then created in which each cell contains the number of times any word used in the corresponding category co-occurs with the corresponding word. The co-occurred concepts are added to the distributional profile of the input keyword.

The semantic distance of two concepts is defined as the cosine coefficient of their distributional profiles:

$$S_{MH}(c_1, c_2) = \frac{\sum_{w \in C(c_1)+C(c_2)} P(w|c_1) \times P(w|c_2)}{\sqrt{\sum_{w \in C(c_1)} P(w|c_1)^2} \times \sqrt{\sum_{w \in C(c_2)} P(w|c_2)^2}}, \tag{3.19}$$

where $C(x)$ is the set of words that co-occur with concept x within a user-defined window.

3.1.3.5 Li et al.'s Model

Li et al. [36, 37] propose a hybrid method based on lexical information in WordNet and statistics from the Brown corpus to measure the semantic similarity of short texts of sentence length. Their approach incorporates semantic similarity between words, semantic similarity between sentences, and word order similarity to measure the overall sentence similarity.

3.1.3.5.1 Semantic similarity between words

The semantic similarity between two words is a function of their path length and depth of their lowest common ancestor in the WordNet lexical database.

3.1.3.5.2 Semantic similarity between sentences

Given two sentences, this module forms a joint word list containing all the distinct words from the two sentences. A vector of the same length as the joint word list is constructed for each of the two sentences. Each entry in the vector is a weight of the corresponding word from the joint word list. If a word in the joint word list exists in the sentence, its weight is 1, otherwise, the weight is the maximum semantic similarity score between this word and all the words in the sentence.

Next, each of the two vectors is reweighed taking the *information content* of their words in the Brown corpus into account. The semantic similarity between the two sentences is the cosine coefficient of their reweighed vectors.

3.1.3.5.3 Word order similarity between sentences

Li et al.'s model includes an optional module to take account of the similarity of word orders. When this module is enabled, phases such as "a dog bites a man" and "a man bites a dog" are considered different even that they share the same words. Given two sentences, each word in the sentence is assigned a number representing its position. The word order similarity between sentences is determined by the normalized difference in word orders.

3.1.3.5.4 Overall sentence similarity

The overall sentence similarity is a weighted summation of the semantic similarity and the word order similarity between sentences.

3.1.3.6 Ponzetto and Strub's Wikipedia-based Model

Ponzetto and Strub's model [53] computes semantic relatedness between two terms over the Wikipedia (Section 2.3.1) category network. Their method contains four steps:

1. Given two terms t_1 and t_2, retrieve two distinct Wikipedia pages p_1 and p_2 that refer to t_1 and t_2.

2. Connect to the Wikipedia category network by parsing the pages and extracting the two sets of categories the pages belong to.

3. Compute the paths between all pairs of categories of the two pages.

4. Compute semantic relatedness based on the two pages extracted (for text overlap based measures) and the paths found along the category network (for path length and information content based measures).

The information content (IC) of a category node n in the hierarchy is a function of its child nodes:

$$IC(n) = 1 - \frac{\log(hypo(n) + 1)}{\log(C)}, \tag{3.20}$$

where $hypo(n)$ is the number of hyponyms of node n and C is the total number of nodes in the hierarchy.

The semantic relatedness function of two terms t_1 and t_2 is based on the overlap percentage of their corresponding pages p_1 and p_2:

$$S_{PS}(t_1, t_2) = \tanh\left(\frac{overlap(p_1, p_2)}{length(p_1) + length(p_2)}\right), \tag{3.21}$$

where $overlap(p_1, p_2)$ is the overlap score [35] of pages p_1 and p_2, and $length(p)$ is the document length of page p. The hyperbolic tangent is used to ensure the output is within $[0, 1]$.

3.2 Wikipedia for Word Sense Disambiguation

Mihalcea and Csomai [43] introduce the system Wikify! which uses Wikipedia as a resource for automatic keyword extraction and word sense disambiguation. Wikify! accepts a news article as input, and identifies the important concepts in the text using keyword extraction. With word sense disambiguation built on top of the existing Wikipedia annotations (as represented in the Wikipedia page titles), Wikify! links the identified concepts to the Wikipedia articles that most likely correspond to the correct senses.

CHAPTER 4

A GENERIC APPROACH FOR COURSES FROM MULTIPLE MAJORS

4.1 Proposed Method

This section proposes a variant of the hybrid method by Li et al. [37] to identify course equivalencies by measuring the semantic relatedness between course descriptions. The approach has three modules: (1) semantic relatedness between words, (2) semantic relatedness between sentences, and (3) semantic relatedness between paragraphs. This work modifies the semantic similarity between words and the semantic similarity between sentences modules developed by Li et al. and adds semantic relatedness between paragraphs tailored to the domain of identifying equivalent courses [67]. Experiments show that these modifications improve the accuracy compared to related work.

4.1.1 Semantic Relatedness Between Words

Given a concept c_1 of word w_1, and a concept c_2 of word w_2, the *semantic relatedness between the words* (SRBW) is a function of the path length between the two concepts and the depth of their lowest common hypernym.

The path length p from c_1 to c_2 is determined by one of five cases. This work adds holonymy and meronymy relations to the method by Li et al. [37] to measure the semantic relatedness:

1. c_1 and c_2 are in the same synonym set (synset).

2. c_1 and c_2 are not in the same synset, but the synset of c_1 and the synset of c_2 contain one or more common words.

3. c_1 is either a holonym or a meronym of c_2.

4. c_1 is neither a holonym nor a meronym of c_2, but the synset of c_1 contains one or more words that are either holonyms or meronyms of one or more words in the synset that c_2 belongs to.

5. c_1 and c_2 do not satisfy any of the previous four cases.

If c_1 and c_2 belong to case 1, p is 0. If c_1 and c_2 belong to cases 2, 3, or 4, p is 1. In case 5, p is the number of links between the two words. The semantic relatedness of c_1 and c_2 is an exponential decaying function of p, where α is a constant [37]:[1]

$$f_1(p) = e^{-\alpha p} \quad (\alpha \in [0,1]). \tag{4.1}$$

Let h be the depth of the lowest common hypernym of c_1 and c_2 in the WordNet hierarchy. f_2 is a monotonically increasing function of h [37]:

$$f_2(h) = \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \quad (\beta \in [0,1]). \tag{4.2}$$

The semantic relatedness between concepts c_1 and c_2 is defined as:

$$f_{word}(c_1, c_2) = f_1(p) \cdot f_2(h), \tag{4.3}$$

where f_1 and f_2 are given by Equations 4.1 and 4.2. The values of both f_1 and f_2 are between 0 and 1 [37].

WordNet is based on concepts, not words. *Unigrams* with different meanings are considered different words and are marked with sense tags [9]. Unfortunately,

[1] In the experiment, $\alpha = -0.2$ and $\beta = 0.45$.

common corpora (as well as course descriptions) are not sense-tagged. Therefore, a mapping between a word and a certain sense must be provided. Such a mapping is formally known as *word sense disambiguation* (WSD), which is the ability to identify the meaning of words in context in a computational manner [50]. This work considers two strategies to perform the WSD: (1) compare all senses of two words and select the maximum score, and (2) apply the first sense heuristic [41].[2] The experiment will compare the performance of these WSD strategies.

To improve accuracy, the *parts of speech* (POS, see Appendix A) of two words have to be the same before visiting the WordNet taxonomy to determine their semantic relatedness. We consider "book" as in "read a book" and "book" as in "book a ticket" to be different. We do not distinguish the plural forms of POS from singular forms. POS such as "NN" (the singular form of a noun) and "NNS"(the plural form of a noun) are therefore considered the same.

The SRBW module also considers the *stemmed* forms of words. Without considering stemmed words, two equivalent course titles such as "networking" and "data communication" are misclassified as semantically distant because "networking" in WordNet is solely defined as socializing with people, not as a computer network. The stemmed word "network" is semantically closer to "data communication."

Algorithm 1 shows how to determine the semantic relatedness between two words w_1 and w_2.

The SRBW module uses WordNet as a lexical knowledge base to determine the semantic closeness between words. The path lengths and depths in the WordNet IS_A hierarchy may be used to measure how strongly a word contributes to the meaning of a sentence. However, this approach has a problem. As mentioned previously, WordNet suffers the knowledge acquisition bottleneck (Section 1.2). Because WordNet is a

[2] The first sense heuristic always selects the first sense of a polysemous word in a hierarchy.

Algorithm 1 Semantic Relatedness Between Words

1: If two words w_1 and w_2 have different POS, consider them semantically distant. Return 0.
2: If w_1 and w_2 have the same spelling and the same POS but do not exist in WordNet, consider them semantically close. Return 1.
3: Using either maximum scores or the first sense heuristic to perform WSD, measure the semantic relatedness between w_1 and w_2 using Equation 4.3.
4: Using the same WSD strategy as the previous step, measure the semantic relatedness between the stemmed w_1 and the stemmed w_2 using Equation 4.3.
5: Return the larger of the two results in steps (3) and (4), i.e., the score of the pair that is semantically closer.

manually created lexical resource, it does not cover all the words that appear in a sentence, even though some of these words are commonly seen in literature. Words not defined in WordNet are misclassified as semantically distant when compared with any other words (unless they have the same spelling and same POS). This is a huge problem for identifying equivalent courses. For example, course names "propositional logic" and "logic" are differentiated solely by the word "propositional," which is not defined in WordNet[3]. The semantic relatedness measurement between *sentences* therefore cannot be simplified to all pairwise comparisons of words using WordNet. A corpus must be introduced to assess the importance of words in sentences.

4.1.2 Semantic Relatedness Between Sentences

To measure the semantic relatedness between sentences, Li et al. [37] join two sentences S_1 and S_2 into a unique word set S, with a length of n (Section 3.1.3.5.2):

$$S = S_1 \cup S_2 = \{w_1, w_2, \dots w_n\}. \tag{4.4}$$

A semantic vector SV_1 is computed for sentence S_1 and another semantic vector SV_2 for sentence S_2. Given the number of words in S_1 as t, Li et al. [37] define the value

[3]WordNet 3.0 was used in the implementation and experiments.

of an entry of SV_1 for sentence S_1 as:

$$SV_{1i} = \hat{s_{1i}} \cdot I(w_i) \cdot I(w_{1j}), \tag{4.5}$$

where $i \in [1, n]$, $j \in [1, t]$, $\hat{s_{1i}}$ is an entry of the lexical semantic vector $\hat{s_1}$ derived from S_1, w_i is a word in S, and w_{1j} is semantically the closest to w_i in S_1. $I(w_i)$ is the information content (IC) of w_i in the Brown corpus and $I(w_{1j})$ is the IC of w_{1j} in the same corpus.

Our work redefines the i-th component of the semantic vector as:

$$SV_{1i} = \hat{s_{1i}} \cdot (\text{TF-IDF}(w_i) + \epsilon) \cdot (\text{TF-IDF}(w_{1j}) + \epsilon). \tag{4.6}$$

There are two major modifications in our version compared to Li et al.'s work. First, we replace the information content with the *TF-IDF* weighting scheme (Section 3.1.2.1), which is a bag-of-words model [30]. The TF-IDF weight of the i-th term (t_i) in document D is a product of the term frequency and the inverted document frequency (Equation 3.8).Our approach uses a smoothing factor ϵ to add a small mass[4] to the TF-IDF.

Second, TF-IDF is computed over the custom course description corpus instead of the Brown corpus. The course description corpus is built from crawling the course catalogs from two universities' websites. These two modifications look for inner relations of words from the course description data domain, rather than from the various domains provided by the Brown corpus.

The first-level semantic relatedness of S_1 and S_2, namely $f^{(1)}_{sent}(S_1, S_2)$ is the cosine coefficient of their semantic vectors SV_1 and SV_2 [37]:

$$f^{(1)}_{sent}(S_1, S_2) = \frac{SV_1 \cdot SV_2}{||SV_1|| \cdot ||SV_2||}. \tag{4.7}$$

[4]In our experiments, ϵ=0.01.

Although Li et al. [37] do not remove *stop words*,[5] we found that the removal of stop words remarkably improves accuracy to identify equivalent courses. (See Section 4.2.)

Algorithm 2 Lexical Semantic Vector $\hat{s_1}$ for S_1

1: **for all** words $w_i \in S$ **do**
2: if $w_i \in S_1$, set $\hat{s_{1i}} = 1$ where $\hat{s_{1i}} \in \hat{s_1}$.
3: if $w_i \notin S_1$, the semantic relatedness between w_i and each word $w_{1j} \in S_1$ is calculated (Section 4.1.1). Set $\hat{s_{1i}}$ to the highest score if the score exceeds a preset threshold δ ($\delta \in [0,1]$), otherwise $\hat{s_{1i}} = 0$.
4: Let $\gamma \in [1, n]$ be the maximum number of times a word $w_{1j} \in S_1$ is chosen as semantically the closest word of w_i. Let the semantic relatedness of w_i and w_{1j} be d, and f_{1j} be the number of times that w_{1j} is chosen. If $f_{1j} > \gamma$, set $\hat{s_{1i}} = d/f_{1j}$ to give a penalty to w_{1j}. This step is called *ticketing*.
5: **end for**

While building and deriving the lexical semantic vectors $\hat{s_1}$ for sentence S_1 and $\hat{s_2}$ for sentence S_2, we found that some words from the joint word list S (Equation 4.4) which are not stop words, but are very generic, in turn rank as semantically the closest words to most other words. These generic words cannot be simply regarded as domain-specific stop words in that a generic word in a pair of courses may not be generic in another pair. To discourage these generic words, we introduce a *ticketing algorithm* as part of the process to build a lexical semantic vector. Algorithm 2 shows the steps to build the lexical semantic vector[6] $\hat{s_1}$ for sentence S_1. Similarly, we follow these steps to build $\hat{s_2}$ for S_2.

The approach proposed by Li et al. [37] contains an optional module that measures *word order similarity*. Each word in the unique word list S (Equation 4.4) is assigned a unique number. Two word order vectors Q_1 and Q_2 are created from S_1 and S_2. Each entry in Q_1 is the assigned number in S of the corresponding word in S_1. Q_2 is

---

[5] Stop words (such as "the", "a", and "of") are words that appear in almost every document, and have no discrimination value for contexts of documents. Porter et al.'s English stop words list (`http://snowball.tartarus.org/algorithms/english/stop.txt`) was adapted for this work.

[6] In our experiments, we chose δ=0.2.

created similarly. The word order similarity of S_1 and S_2 is the normalized difference of their word order vectors [37]:

$$f_{order}(S_1, S_2) = 1 - \frac{||Q_1 - Q_2||}{||Q_1 + Q_2||}. \tag{4.8}$$

The second-level semantic relatedness of sentences S_1 and S_2 combines the first-level semantic relatedness and the word order similarity:

$$f^{(2)}_{sent}(S_1, S_2) = \tau \cdot f^{(1)}_{sent}(S_1, S_2) + (1 - \tau) \cdot f_{order}(S_1, S_2), \quad \tau \in [0, 1]. \tag{4.9}$$

4.1.3 Semantic Relatedness Between Paragraphs

Although Li et al. [37] claim that their approach is for measuring the semantic similarity of sentences and short texts, preliminary experiments show that the accuracy of their approach is not satisfactory on course descriptions. This section introduces the semantic relatedness measure between paragraphs to address the problem.

Given two course abstracts P_1 and P_2, the first step is to remove generic data and prerequisite information. Let P_1 be a paragraph consisting of a set of n sentences, and P_2 be a paragraph of m sentences, where n and m are positive integers. For s_{1i} ($s_{1i} \in P_1$, $i \in [1, n]$) and s_{2j} ($s_{2j} \in P_2$, $j \in [1, m]$), the semantic relatedness between paragraphs P_1 and P_2 is defined as a weighted mean:

$$f_{para}(P_1, P_2) = \frac{\sum_{i=1}^{n} (\max_{j=1}^{m} f^{(2)}_{sent}(s_{1i}, s_{2j})) \cdot N_i}{\sum_{i=1}^{n} N_i}, \tag{4.10}$$

where N_i is the sum of the number of words in sentences s_{1i} ($s_{1i} \in P_1$) and s_{2j} ($s_{2j} \in P_2$), and $f_{sent}(s_{1i}, s_{2j})$ is the semantic relatedness between sentences s_{1i} and s_{2j} (Section 4.1.2). Algorithm 3 summarizes these steps. Optionally, the *deletion* flag can be enabled to speed up the computation. Empirical results show that accuracy is about the same whether or not the *deletion* flag is enabled.

Algorithm 3 Semantic Relatedness for Paragraphs

1: If *deletion* is enabled, given two course abstracts, select the one with fewer sentences as P_1, and the other as P_2. If *deletion* is disabled, select the first course abstract as P_1, and the other as P_2.
2: **for** each sentence $s_{1i} \in P_1$ **do**
3: Calculate the semantic relatedness between sentences (Section 4.1.2) for s_{1i} and each of the sentences in P_2.
4: Find the sentence pair $\langle s_{1i}, s_{2j} \rangle$ ($s_{2j} \in P_2$) that scores the highest. Save the highest score and the total number of words of s_{1i} and s_{2j}. If *deletion* is enabled, remove sentence s_{2j} from P_2.
5: **end for**
6: Collect the highest score and the number of words from each run. Use their weighted mean (Equation 4.10) as the semantic relatedness between P_1 and P_2.

Given title T_1 and abstract P_1 of course C_1, and title T_2 and abstract P_2 of course C_2, the semantic relatedness of the two course descriptions is defined as:

$$f_{course}(C_1, C_2) = \theta \cdot f^{(2)}_{sent}(T_1, T_2) + (1 - \theta) \cdot f_{para}(P_1, P_2), \quad \theta \in [0, 1]. \tag{4.11}$$

Parameter θ denotes how much course titles weigh over course abstracts. Course titles are compared using the semantic relatedness measurement discussed in Section 4.1.2, and course abstracts are compared using the measure discussed in Section 4.1.3.

4.2 Implementation and Experimental Results

The method proposed in this section is fully implemented using Python and the Python *Natural Language Toolkit* (NLTK) [4].[7] The WordNet interface built into NLTK is used to retrieve lexical information for word similarities. We use the following default parameters in our experiments: $\alpha = -0.2$ (Equation 4.1), $\beta = 0.45$ (Equation 4.2), $\tau = 0.85$ (Equation 4.9), $\delta = 0.2$ (Algorithm 2), $\gamma = 2$ (Algorithm 2), $\theta = 0.7$ (Equation 4.11), and $\epsilon = 0.01$ (Equation 5.2). The α, β, and τ use the recommended

[7]NLTK: http://nltk.org/

setting by Li et al. [37]. We choose the values for δ, ϵ, γ and θ based on empirical results over development data sets.

A course description corpus must be built for the experiments. The UML course transfer dictionary lists courses that are equivalent to those from hundreds of other institutions (Figure 1.1, page 2). We picked Middlesex Community College (MCC) as an external institution in our experiments. The transfer dictionary lists over 1,400 MCC courses in different majors. We remove the rejected courses, elective courses, and those with missing fields from the transfer dictionary. Referring to the equivalencies from the transfer dictionary, we crawl over 1,500 web pages from the course catalogs of both UML and MCC to retrieve over 200 interconnected courses that contain both course names and descriptions. Next, we created two XML files, one for UML and one for MCC courses. Given an MCC course, the goal is to suggest the most similar UML course. A fragment of the MCC XML file is shown below. Each course entry has features such as course ID, course name, credits, description, and the ID of its equivalent course at UML. The UML XML file has the same layout except that the *equivalence* tag is removed and the root tag is *uml*. Each MCC course is compared to all the UML courses and the *equivalence* tag in the MCC XML file is used as the "ground truth" validation.


```
<mcc>
  <course>
    <courseid>ART 113</courseid>
    <coursename>Color and Design</coursename>
    <credits>3</credits>
    <description>Basic concepts of composition
    and color theory. Stresses the process and
    conceptual development of ideas in two
    dimensions and the development of a strong
```

```
    sensitivity to color.</description>
    <equivalence>70.101</equivalence>
  </course>
  ...
</mcc>
```

After the integrity check, the MCC XML file contains 108 courses and the UML XML file contains 89 courses. The reason there are more MCC courses than UML courses is that the transfer dictionary allows multiple courses from MCC to be transferred to the same UML course.

To monitor the accuracy change over different numbers of documents, we randomly select equivalent courses to create two smaller data sets for UML and MCC respectively in the XML format. The random number of courses in each XML file is shown in Table 4.1. These three pairs of XML data sets are used both as the corpora and as the test data sets.

XML Data Sets	MCC Courses	UML Courses	Total
Small	25	24	49
Medium	55	50	105
Large	108	89	197

Table 4.1: Number of courses in the data sets

Consider the small data set as an illustration. Each of the 25 MCC courses is compared with all 24 UML courses. All words are converted to lowercase and punctuation is removed. We also remove both *general stop words*[8] (such as "a" and "of") and *domain-specific stop words*[9] (such as "courses," "students," and "reading"). We do not remove words based on high or low occurrences because our preliminary

[8]The experiments use the Snowball [54] English stop word list: `http://snowball.tartarus.org/algorithms/english/stop.txt`.

[9]A list of domain-specific stop words is created manually.

experiments found empirically that this *decreases* accuracy. Using the algorithms discussed in Section 4.1, a score is computed for each comparison. After comparing an MCC course to all UML courses, the 24 UML courses are sorted by score in descending order. The course equivalencies indicated by the transfer dictionary are used as the benchmark. In each run we mark the rank of the real UML course that is equivalent to the given MCC course as indicated by the transfer dictionary. We consider the result of each run correct when the equivalent course indicated by the transfer dictionary is in the top 3 of the sorted list.[10] After doing this for all of the 25 MCC courses, we calculate the overall accuracy and the average ranks of the real equivalent courses.

Figure 4.1 and Figure 4.2 report such results. Note that for any of the algorithms, both accuracy and average rank decrease as the number of documents increases. The more documents the algorithm is experimented on, the more likely it would run into the sparsity issue in WordNet.

For each of the three different approaches, we note the average ranks of the real equivalent courses indicated by the transfer dictionary. Figure 4.2 shows that our approach outperforms the TF-IDF and Li et al. [37] approaches. It also shows that the performance is better when the word order similarity is enabled.

Both accuracy (Figure 4.1) and rank (Figure 4.2) suggest performance is slightly better when the word order similarity is enabled. As the number of documents increases, enabling word order has fewer advantages than disabling it. In addition, it takes at least twice long to run when the word order similarity is enabled. When efficiency is a high priority, doubling the amount of time to achieve only a small degree of performance improvement does not appear to be worthwhile.

This work considers two strategies to perform the WSD: (1) compare all senses

[10] Top 3 is chosen instead of top 1 because the UML transfer dictionary allows multiple courses from an external institution to be transferred to the same course offered at UML.

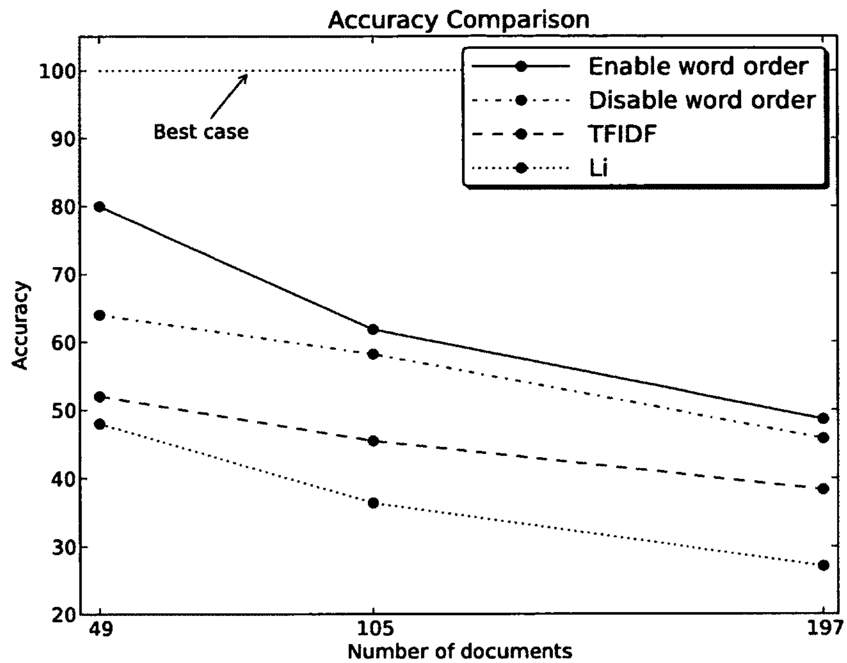


Figure 4.1: Accuracy of our approach compared to the TF-IDF and Li et al. [37] approaches.

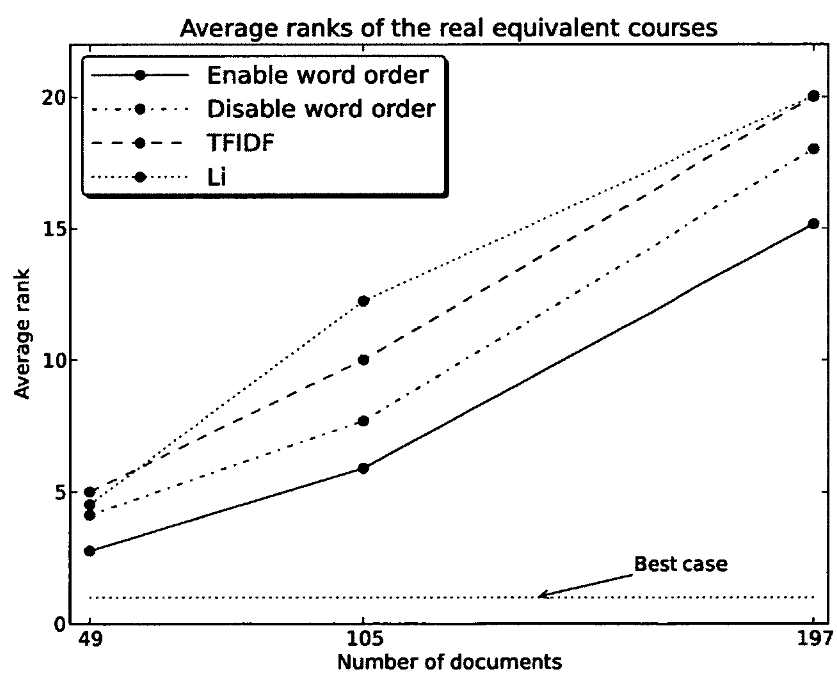


Figure 4.2: Average ranks of the real equivalent courses.

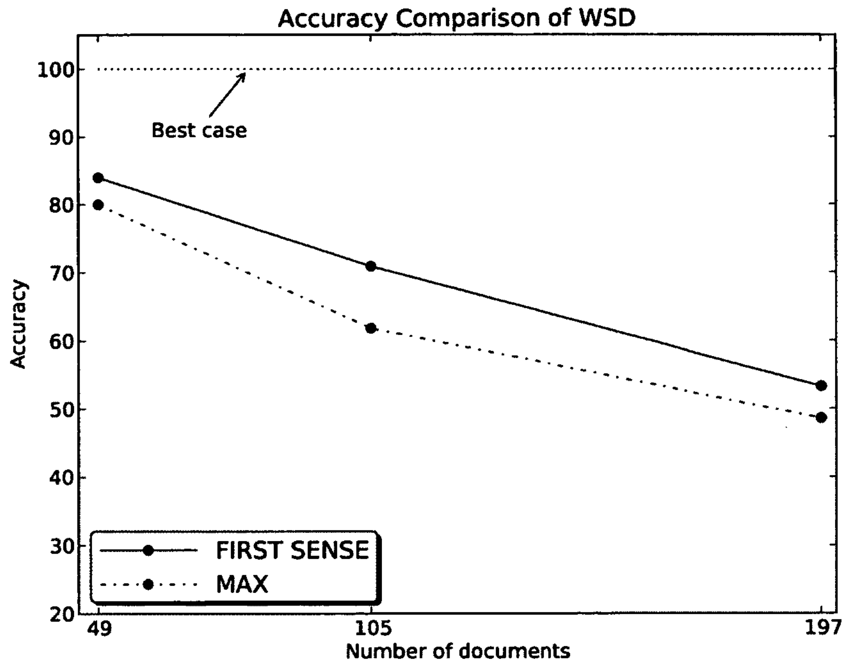


Figure 4.3: Accuracy of the two WSD strategies.

of two words and select the maximum score (MAX), and (2) apply the first sense heuristic [41] (FIRST SENSE). Figure 4.3 shows the accuracies of the two WSD strategies. The first sense heuristic performs better than selecting the maximum score over the three pairs of data sets from Table 4.1.

4.3 Conclusion

This chapter presents a novel application of semantic relatedness to suggesting potential equivalencies for a course transferred from an external university. It proposes a hybrid method that incorporates semantic relatedness measurement for words, sentences, and paragraphs. We show that a composite weighting scheme based on a lexicographic resource and a bag-of-words model outperforms previous work to identify equivalent courses. By enabling word order similarity it takes twice the amount of time to run and the performance is only slightly improved. Therefore in our experiments, word order similarity is not very useful for identifying course equivalencies.

Most of the courses in our data set are from liberal arts. WordNet as a knowledge source is sufficient for these courses, since they do not contain many technical terms. The next chapter will reveal that WordNet is not an ideal choice for suggesting equivalent courses from technical fields of study. We will propose a new approach to address the problem.

CHAPTER 5

A DOMAIN-SPECIFIC APPROACH FOR COURSES FROM ONE MAJOR

5.1 What's Wrong with WordNet?

Traditional knowledge bases (such as WordNet) suffer the *knowledge acquisition bottleneck* (Section 1.2, page 3). As a result, most of the technical terms are missing in such a knowledge base. These technical terms are crucial to help determine the equivalencies of technical courses that are packed with such terms. To illustrate, consider the following course:

> **91.304 Foundations of Computer Science**: A survey of the mathematical foundations of Computer Science. Finite automata and regular languages. Stack Acceptors and Context-Free Languages. Turing Machines, recursive and recursively enumerable sets. Decidability. Complexity. This course involves no computer programming.

The following 64 *unfiltered* WordNet synsets are retrieved by querying WordNet with the *n-grams* ($n = \{1, 2, 3\}$) generated from the course description shown above:

> acceptor, adjust, arrange, automaton, basis, batch, bent, calculator, car, class, complexity, computer, countable, course, determine, dress, even, finite, fix, foundation, foundation garment, fructify, hardening, imply, initiation, involve, jell, language, linguistic process, lyric, machine, mathematical, naturally, necessitate, numerical, path, place, plant, push-down list, push-down storage, put, recursive, regular, review, rig, run, science,

set, set up, sic, sketch, skill, smokestack, specify, speech, stack, stage set, surveil, survey, terminology, turing, typeset, unconstipated, view.

On the other hand, if we use the Wikipedia-based approach outlined in this chapter, the following 18 Wikipedia articles are retrieved:

> Alan Turing, Algorithm, Automata theory, Complexity, Computer, Computer science, Context-free language, Enumeration, Finite set, Finite-state machine, Kolmogorov complexity, Language, Machine, Mathematics, Recursive, Recursive language, Recursively enumerable set, Set theory.

Although the WordNet-based approach generates more features from the given course description, the Wikipedia-based approach captures information more precisely.[1] In addition, the WordNet-based approach produces more noise. For example, it interprets word "automata" in "finite automata" as "automaton," and word "regular" in "regular languages" as "unconstipated."

As the example above shows, a semantic relatedness measure based on Wikipedia is likely to be more accurate to match equivalent courses from fields that are heavily equipped with technical terms when compared to other measures based on a traditional knowledge base such as WordNet. Although it started over 10 years later than WordNet, Wikipedia has grown to be much larger (Figure 5.1). This chapter proposes a domain-specific semantic relatedness measure that analyzes course descriptions to suggest whether a course can be transferred from one institution to another. Wikipedia is chosen as the knowledge base due to its rich contents (Figure 5.2) and continuously coalescent growth [6]. In addition, the multilingual Wikipedia makes it easy to adapt this work to suggest equivalencies for courses in other natural languages.

The proposed approach is different from related work [43, 53, 22] using Wikipedia. While Mihalcea and Csomai [43] use the annotation in the page title of a concept to

[1]The comparison is based on WordNet 3.0 and Wikipedia of July 2011.

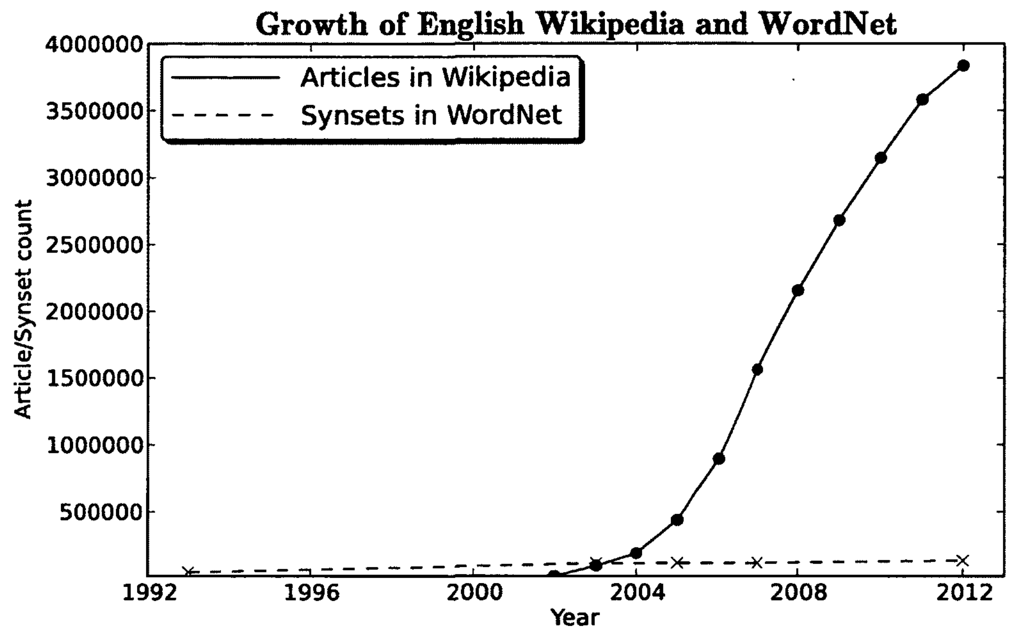


Figure 5.1: Growth of Wikipedia and WordNet over the years

perform WSD (Section 3.2), the proposed approach uses a page's parent category as a cue to the correct sense. Ponzetto and Strube [53] limit their measurement to word pairs (Section 3.1.3.6), while this study focuses on text of any length. Gabrilovich and Markovitch [22] compute TF-IDF statistics for every word and every document of Wikipedia (Section 3.1.2.5) which is highly inefficient. They also remove category pages and disambiguation pages. In contrast, the proposed model is mainly based on the category taxonomy and the corpus statistics are limited to metadata that are mostly available in Wikipedia. Furthermore, we compute concept relatedness on a domain-specific hierarchy that weighs both path lengths and diversions from the topic. The domain-specific hierarchy is much smaller than the entire Wikipedia corpus. As a result, the proposed algorithm is more efficient than previous work.

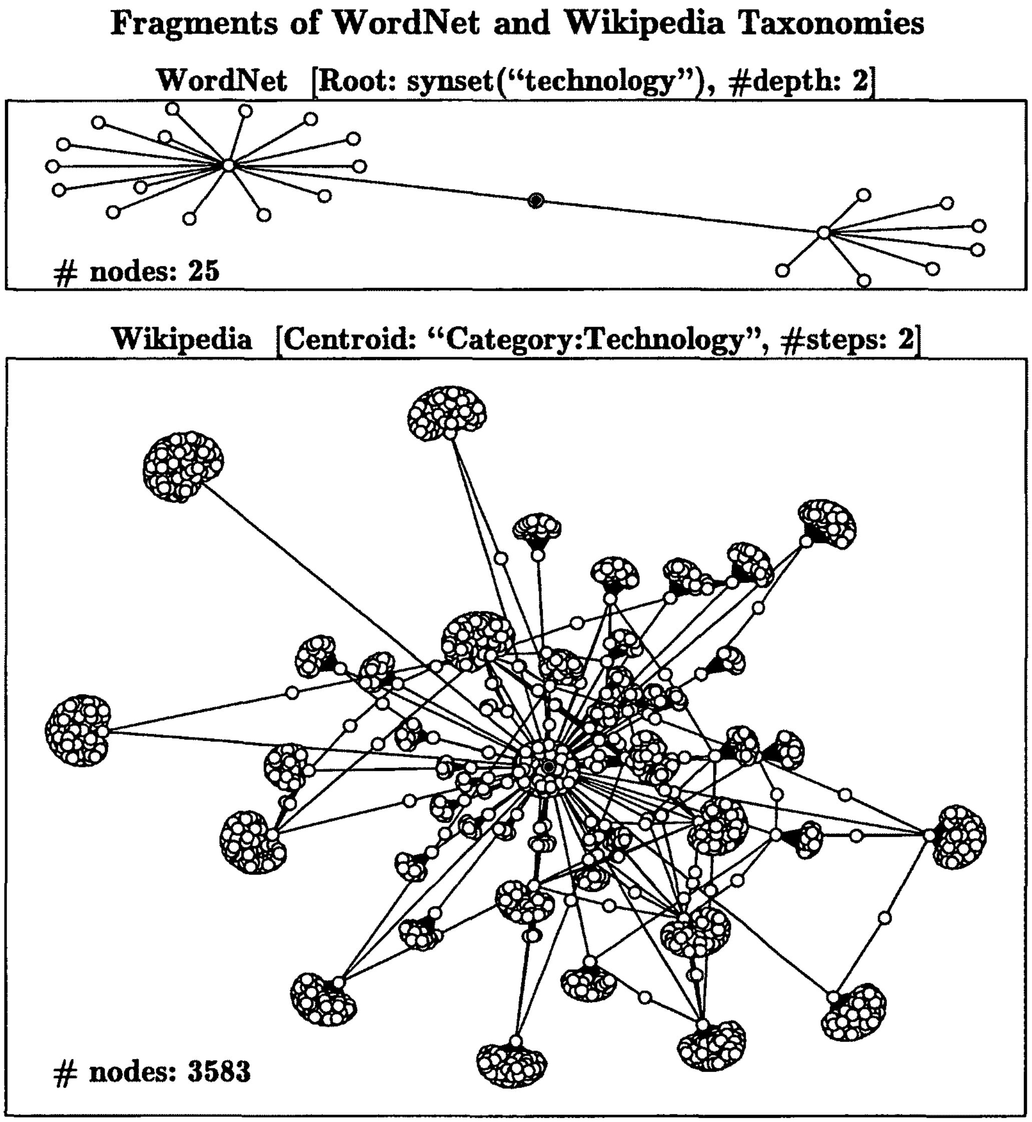


Figure 5.2: Fragments of WordNet 3.0 (top) and English Wikipedia of 2011/7 (bottom) taxonomies. The root/centroid node is shown in red and is located at the very center of each figure.

5.2 Proposed Method

The proposed method contains four modules. Section 5.2.1 explains how to construct a domain-specific hierarchy from Wikipedia. Section 5.2.2 presents semantic relatedness between concepts. Section 5.2.3 describes the steps to generate features from course descriptions. And Section 5.2.4 evaluates course relatedness.

5.2.1 Extract a Lexicographical Hierarchy from Wikipedia

When a domain is specified (e.g., CS courses), we start from a generic Wikipedia category in this domain, choose its parent as the root, and use a depth-limited search to recursively traverse each subcategory (including subpages) to build a lexicographical hierarchy with depth D. For example, to find CS course equivalencies, we built a hierarchy using the parent of "Category:Computer science," i.e., "Category:Applied sciences," as the root. We choose the parent of the generic category as the root to make sure the hierarchy not only covers the terms in this domain, but also those in neighbor domains. The hierarchy of "Category:Applied sciences" not only covers Computer Science, but also related fields such as Computational Linguistics and Mathematics.

Depth (D)	Number of Concepts at this Depth
1	71
2	4,177
3	60,158
4	177,955
5	494,039
6	1,848,052

Table 5.1: Number of concepts at each depth in the "Category:Applied sciences" hierarchy.

Table 5.1 reports the number of nodes per level for the hierarchy of applied sciences. Figure 5.3 visualizes the growth of this hierarchy as the depth increases from 1 to 3. Both the number of nodes and number of edges in the hierarchy grow ex-

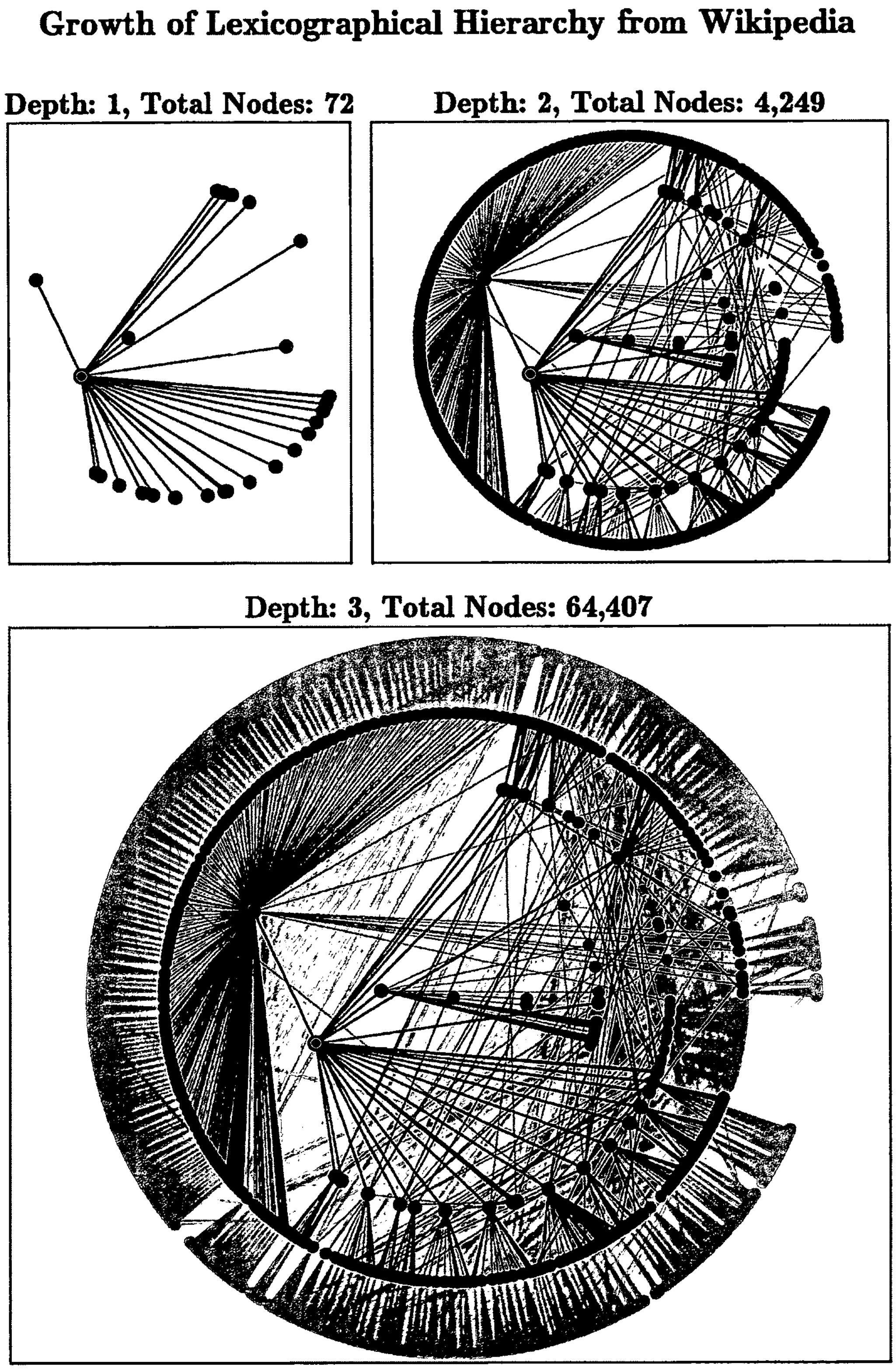


Figure 5.3: Growth of the lexicographical hierarchy constructed from Wikipedia, illustrated in circular trees. A lighter color of the nodes and edges indicates that they are at a deeper depth in the hierarchy.

ponentially as the depth increases. Therefore, D need not be a big number to cover most terms in the domain. We have found the hierarchy speeds up the semantic measurement dramatically and covers almost all the words in the specific domain. In the experiment on CS courses (D=6), we eliminated over 71% of Wikipedia articles,[2] yet the hierarchy covered almost all the important terms mentioned in the course descriptions.

5.2.2 Semantic Relatedness Between Concepts

Similar to the work of Li et al. [37] and the first proposed approach (Equation 4.3), the semantic relatedness between two Wikipedia concepts,[3] t_1 and t_2 in the hierarchy is defined as:

$$f'(t_1, t_2) = e^{-\alpha p} \cdot \frac{e^{\beta d} - e^{-\beta d}}{e^{\beta d} + e^{-\beta d}} \quad (\alpha, \beta \in [0, 1]), \tag{5.1}$$

where p is the shortest path between t_1 and t_2, and d is the depth of the lowest common hypernym of t_1 and t_2 in the hierarchy (Section 5.2.1). This is different from related work on semantic relatedness from Wikipedia [53] in that we not only consider the shortest path (p) between two concepts but also their common distance (d) from the topic, which in turn emphasizes domain awareness.

5.2.3 Generate Course Description Features

The built-in redirection in Wikipedia is useful for spelling corrections because variations of a term redirect to the same page. To generate features from a course description C, we start by generating *n-grams* ($n \in [1, 3]$) from C. We then query the *redirection data* to fetch all pages that match any of the *n-grams*.

---

[2]The hierarchy contains 1,534,267 distinct articles, as opposed to 5,329,186 articles in Wikipedia.

[3]Each concept corresponds to a Wikipedia page.

The identified pages are still sparse. We therefore query the *title data* to fetch those that match any of the *n-grams*. Page topics are not discriminated in this step. For example, *unigram* "Java" returns both "Java (software platform)" and "Java (dance)."

Wikipedia contains a collection of disambiguation pages. Each disambiguation page includes a list of alternative uses of a term. Note that there are two different Wikipedia disambiguation pages: *explicit* and *implicit*. A page is *explicit* when the page title is annotated by Wikipedia as "disambiguation," such as "Oil (disambiguation)." A page is *implicit* when it is *not* so annotated, but points to a category such as "Category:Disambiguation pages," or "Category:All disambiguation pages." We iterate over the pages fetched from the last step, using disambiguation pages to enrich and refine the features of a course description.

Unlike the work of Mihalcea and Csomai [43] which uses the annotation in the page title of a concept to perform WSD, the proposed approach uses a page's parent category as a cue to the correct sense. Typically, the sense of a concept depends on the senses of other concepts in the context. For example, a paragraph on programming languages and data types ensures that "data" more likely corresponds to a page under "Category:Computer data" than one under "Category:Star Trek."

Algorithm 4 explains the steps to generate features for a course C.

Given the courses C_1 and C_2 in Chapter 1 (page 3), their generated features F_1 and F_2 are:

> F_1: Shortest path problem, Tree traversal, Spanning tree, Tree, Analysis, List of algorithms, Completeness, Algorithm, Sorting, Data structure, Structure, Design, Data.
>
> F_2: Unix, Social, Ethics, Object-oriented design, Computer programming, C++, Object-oriented programming, Design.

Algorithm 4 Feature Generation (F) for Course C

1: $T_c \leftarrow \emptyset$ (clear terms), $T_a \leftarrow \emptyset$ (ambiguous terms).
2: Generate all possible *n-grams* ($n \in [1,3]$) G from C.
3: Fetch the pages whose titles match any of $g \in G$ from Wikipedia *redirection data.* For each page *pid* of term t, $T_c \leftarrow T_c \cup \{t : pid\}$.
4: Fetch the pages whose titles match any of $g \in G$ from Wikipedia *page title data.* If a disambiguation page, include all the terms this page refers to. If a page *pid* corresponds to a term t that is not ambiguous, $T_c \leftarrow T_c \cup \{t : pid\}$, else $T_a \leftarrow T_a \cup \{t : pid\}$.
5: For each term $t_a \in T_a$, find the disambiguation that is on average most related (Equation 5.1) to the set of clear terms. If a page *pid* of t_a is on average the most related to the terms in T_c, and the relatedness score is above a preset threshold δ ($\delta \in [0,1]$), set $T_c \leftarrow T_c \cup \{t_a : pid\}$. If t_a and a clear term are different senses of the same term, keep the one that is more related to all the other clear terms.
6: Return clear terms as features.

Algorithm 5 Semantic Vector SV_1 for F_1 and J

1: **for all** words $t_i \in J$ **do**
2:     if $t_i \in F_1$, set $SV_{1i} = 1$ where $SV_{1i} \in SV_1$.
3:     if $t_i \notin F_1$, the semantic relatedness between t_i and each term $t_{1j} \in F_1$ is calculated (Equation 5.1). Set SV_{1i} to the highest score if the score exceeds the preset threshold δ, otherwise $SV_{1i} = 0$.
4: **end for**

5.2.4 Determine Course Relatedness

Given two short texts C_1 and C_2, we use Algorithm 4 to generate features F_1 for C_1, and F_2 for C_2. Next, the two feature lists are joined together into a unique set of terms, namely J. Similar to previous work [37], semantic vectors SV_1 (Algorithm 5) and SV_2 are computed for F_1 and F_2.

This work takes into account of the importance of a term by reweighing the semantic vectors using corpus statistics. Each value of an entry of SV_1 for features F_1 is reweighed as:

$$SV_{1i} = SV_{1i} \cdot I(t_i) \cdot I(t_j), \tag{5.2}$$

where SV_{1i} is the semantic relatedness between $t_i \in F_1$ and $t_j \in J$. $I(t_i)$ is the information content of t_i, and $I(t_j)$ is the information content of t_j. Similarly, we reweigh each value for the semantic vector SV_2 of F_2.

The information content $I(t)$ of a term t is a weighted sum[4] of the category information content $I_c(t)$ and the linkage information content $I_l(t)$:

$$I(t) = \lambda \cdot I_c(t) + (1 - \lambda) \cdot I_l(t). \tag{5.3}$$

Inspired by related work [59] (Equation 3.6), the category information content of term t is defined as a function of its siblings:

$$I_c(t) = 1 - \frac{log(siblings(t) + 1)}{log(N)}, \tag{5.4}$$

where $siblings(t)$ is the number of siblings for t on average, and N is the total number of terms in the hierarchy (Section 5.2.1).

The linkage information content is a function of outlinks and inlinks of the page *pid* that t corresponds to:

[4]The experiment uses $\lambda = 0.6$.

$$I_l(t) = 1 - \frac{inlinks(pid)}{MAXIN} \cdot \frac{outlinks(pid)}{MAXOUT}, \tag{5.5}$$

where $inlinks(pid)$ and $outlinks(pid)$ are the numbers of inlinks and outlinks of a page pid. $MAXIN$ and $MAXOUT$ are the maximum numbers of inlinks and outlinks that a page has in Wikipedia. The $MAXIN$ and $MAXOUT$ are based on the entire Wikipedia to avoid the recalculation when the domain changes. This also ensures the maximum linkage information is unbiased. For the July 2011 wikidump, page "Geographic coordinate system" has the most in-links, a total of 575,277. Page "List of Italian communes (2009)" has the most out-links, a total of 8,103.

The semantic relatedness of the two short texts is a cosine coefficient of the two semantic vectors (similar to Equation 4.7):

$$f(C_1, C_2) = \frac{SV_1 \cdot SV_2}{||SV_1|| \cdot ||SV_2||}. \tag{5.6}$$

Let course 1 have title T_1 and description C_1, and course 2 have title T_2 and description C_2, this module first measures the semantic relatedness of T_1 and T_2 and then the relatedness of C_1 and C_2. The semantic relatedness of the two courses is:

$$f(\text{course}_1, \text{course}_2) = \frac{f(T_1, T_2) \cdot (||F_{T1}|| + ||F_{T2}||) + f(C_1, C_2) \cdot (||F_{C1}|| + ||F_{C2}||)}{||F_{T1}|| + ||F_{T2}|| + ||F_{C1}|| + ||F_{C2}||} + \Omega, \tag{5.7}$$

where $f(T_1, T_2)$ is the semantic relatedness score of the two course titles, $f(C_1, C_2)$ is the semantic relatedness score of the two course abstracts, $||F_{Ti}||$ is the number of distinct features in the title of course i ($i = \{1, 2\}$), $||F_{Ci}||$ is the number of distinct features in the description of course i ($i = \{1, 2\}$), and Ω is an optional parameter that considers human decisions and learns from the results of local knowledge.[5]

[5] Although the Ω parameter is not used in the experiment, optionally it could be enabled to emphasize local knowledge.

5.3 Experimental Results

Wikipedia offers its content as database backup dumps (wikidumps) freely available to download. This study uses the July 22, 2011 wikidump of 31 GB extracted from a 7.0 GB compressed file (pages-articles.xml.bz2) obtained from the Wikimedia website.[6] The WikiPrep tool[7] developed by Gabrilovich [22] is used in this work to split the extracted raw XML into several XML files each with a special purpose, such as pages, categories, redirections, links, etc. These separate XML files are imported into MySQL as tables. Table 5.2 shows some statistics of the wikidump of July 22, 2011:

Item	Count
Number of pages and categories	5,329,186
Number of page-category definitions	23,792,229
Number of links	233,167,100
Number of redirections	4,769,252

Table 5.2: Wikidump statistics of July 22, 2011

Using the steps outlined in Section 5.2.1, an additional table is created for the hierarchy with the parent of "Category:Computer science" (i.e. "Category:Applied sciences") as the root to measure computer science course equivalencies. Section 5.2.1 explains why the parent is chosen as the root to build the hierarchy.

The attributes of each table are indexed to speed up queries.

The implemented database design is shown in Figure 5.4. It contains the following tables:

page_category specifies which category or categories a page belongs to.

Column *pid* is the unique identifier of a Wikipedia page, and column

[6]Wikidump of July 22, 2011: `http://dumps.wikimedia.org/enwiki/20110722/`

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

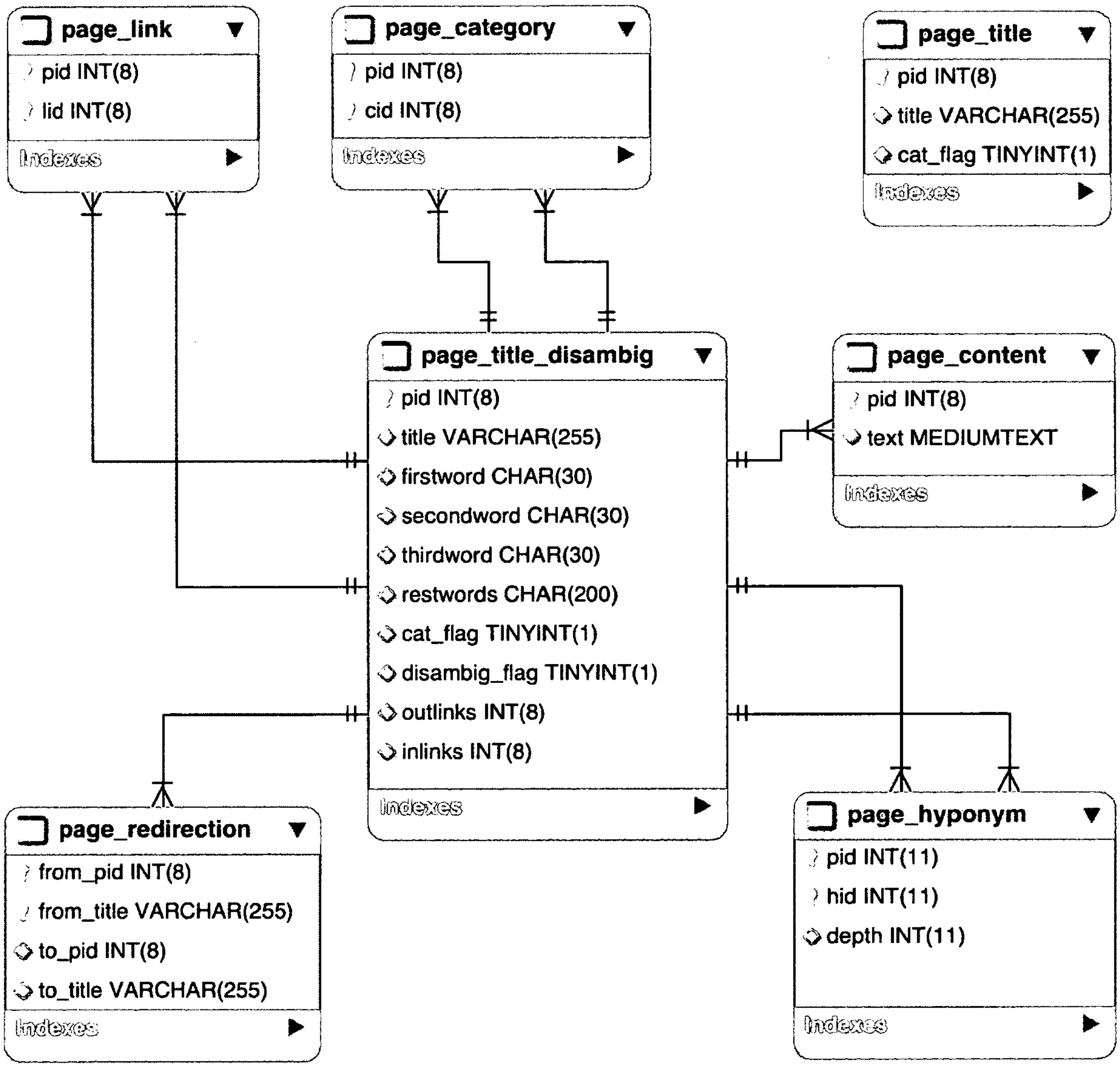


Figure 5.4: Implemented Database Design

cid is a category ID for the page *pid*. This table is extracted and imported from wikidump.

page_content contains the Wikipedia raw page content (*text*) for each page *pid*. This table is extracted and imported from wikidump.

page_redirection lists variations of a term (*from_title*) to their corresponding Wikipedia page (*to_pid* and *to_title*). This table is extracted and imported from wikidump.

page_title lists the unique article identifier (*pid*) and the corresponding article title (*title*) in Wikipedia. This table is extracted and imported from wikidump.

page_title_disambig is built on top of *page_title*. In addition to *pid* and *title*, for each page this table caches the tokenized page title (*firstword*, *secondword*, *thirdword*, and *restwords*), flags to show if the page is a category (*cat_flag*) or a disambiguation page (*disambig_flag*), and statistics of the in-links (*inlinks*) and out-links (*outlinks*) of this page.

page_hyponym contains the domain-specific hierarchy with "Category:Applied sciences" as the root.

Our experiment used $\alpha = 0.2$, $\beta = 0.5$, $\delta = 0.2$, and $\lambda = 0.6$. These values were found empirically to perform well over development data sets. Local knowledge was not used in the experiment ($\Omega = 0$).

Two courses can be considered as equivalent if they are listed as so in the UML course transfer dictionary (Figure 1.1, page 2). However, because the course transfer dictionary is always out of date and it only contains pieces of information about the courses, three problems may arise and in turn affect the accuracy:

1. A pair of courses should be equivalent but the equivalency is not defined in the

UML course transfer dictionary. Missing such data, these courses are unfortunately regarded as not equivalent.

2. An equivalency suggested by the UML course transfer dictionary does not guarantee that the two courses are equivalent. The dictionary is simply a list of course numbers and names that are considered equivalent at the time of evaluation. It does not list course abstracts, which is an important factor to contribute to equivalencies. Course abstracts may change over the years although course numbers do not, and this could affect equivalencies. It is possible an equivalency previously suggested by the transfer dictionary becomes invalid over the time due to the change of course abstracts.

3. An institution may periodically rearrange its catalog and assign its courses different course numbers. An old course number used in the UML course transfer dictionary becomes unrecognized, making the dictionary data more sparse.

Therefore, the traditional precision and recall [64] cannot fit in as evaluation tools. Consequently, this section uses a rank-based scheme to evaluate.

We randomly selected 25 CS courses from 19 universities that can be transferred to University of Massachusetts Lowell (UML) according to the transfer dictionary. Each transfer course was compared to all 44 CS courses offered at UML, a total of 1,100 comparisons. The result was considered correct for each course if the real equivalent course in UML appears among the top 3 in the list of highest scores. We excluded all Wikipedia pages whose titles contained specific dates or were annotated as "magazine," "journal," "book," "dance," "band," "novel," or "album." We removed both general and domain stop words (Appendix B) from course descriptions. If a course description contains the keywords "not" or "no," e.g., "This course requires no computer programming skills," the segment after such keyword is ignored.

The proposed approach is compared against the work by Li et al. [37] and TF-

Algorithm	Accuracy
TF-IDF	32%
Li et al. [37]	52%
Proposed approach (Features)	60%
Proposed approach (Features + IC)	72%

Table 5.3: Accuracy of the proposed method against previous work

IDF on the same data set of course descriptions. Accuracies are reported in Table 5.3. Enabling the information content on top of features in the proposed approach (Features + IC) is able to bring the accuracy from 60% up to 72%. Both versions of the proposed approach have higher accuracies than previous work.

Since the transfer dictionary is always out of date, we found a few equivalent course pairs that were unintuitive. It is necessary to set up a human judgment data set to make a more meaningful evaluation. We first tried the Amazon Mechanical Turk (MTurk)[8] to collect human judgment. A problem set of 1,100 questions (HITs) were posted on MTurk. Each HIT contained a pairs of computer science course descriptions. The MTurk workers were asked to compare these descriptions and to evaluate how much they thought the topics of two course descriptions overlapped. Figure 5.5 shows one of the HITs posted on MTurk. Unfortunately, only 15 questions were evaluated after a week. Most of the results from the workers did not make much sense, even though we only allowed *categorization masters*[9] to evaluate. Mechanical Turk therefore does not seem to be an ideal tool to collect a human judgment data set on course equivalencies, at least not for computer science courses that are packed with technical terms.

[8]Amazon Mechanical Turk: `http://www.mturk.com/`

[9]Categorization masters are elite groups of workers who have demonstrated accuracy on specific types of HITs on the MTurk marketplace. A worker achieves a master distinction by consistently completing HITs of a certain type with a high degree of accuracy across a variety of requesters. Masters must continue to pass Amazon's statistical monitoring to remain MTurk masters.

Compare Course Descriptions

The following are descriptions of two Computer Science courses from different universities. The goal is to help determine whether or not course credits can be transferred between the two courses. Rank the similarity of topics covered in the two courses. Try to compare the meaning instead of strict keyword matching. For example, "C++" and "C++" are 100% similar; "C++" and "Programming language" could be "80%" similar.

First Course:

An integrated symbolic, numerical, and graphical approach to computer problem solving. Structured design; fundamental programming techniques. Computer algebra systems. Scientific, engineering, and mathematical applications.

Second Course:

Development of large software projects. Software engineering principles and practice. Object-oriented analysis and design. CASE productivity aids. Development techniques for program-translation software and web software.

What percentage of the course topics overlap?

- ○ 100%
- ○ 75%
- ○ 50%
- ○ 25%
- ○ 0%

Please provide any comments you may have below, we appreciate your input!

Submit

Figure 5.5: One of the HITs posted on the Mechanical Turk

Alternatively, we asked 6 annotators (UML CS students and professors) to annotate computer science course pairs. Each of the 6 annotators was given a list of 32 pairs of courses with only course titles and descriptions. They independently evaluated whether each pair is equivalent on a scale from 1 to 5.[10] We averaged their evaluations for each pair and converted the scale from [1,5] to [0,1]. (This human judgment data set is reported in Appendix C.) Next, the proposed approach, the work by Li et al. [37], and TF-IDF were tested on the same 32 course pairs. Table 5.4 and 5.5 report Spearman's and Pearson's correlation coefficients of course relatedness scores with human judgment, and statistical significances. For the proposed approach, the correlation and p-value are slightly better when the information content is enabled. Both versions of the proposed approach have higher correlations to the human judgment data set compared to previous work. Furthermore, a smaller p-value indicates the proposed approach is more likely to correlate with human judgment.

Algorithm	Spearman's correlation	p-value
TF-IDF	0.644	$7.00 \cdot 10^{-5}$
Li et al. [37]	0.644	$7.05 \cdot 10^{-5}$
Proposed approach (Features)	0.815	$1.33 \cdot 10^{-8}$
Proposed approach (Features + IC)	0.821	$8.39 \cdot 10^{-9}$

Table 5.4: Spearman's correlation of course relatedness scores with human judgments.

Algorithm	Pearson's correlation	p-value
TF-IDF	0.730	$2 \cdot 10^{-6}$
Li et al. [37]	0.570	0.0006
Proposed approach (Features)	0.845	$1.13 \cdot 10^{-9}$
Proposed approach (Features + IC)	0.851	$6.65 \cdot 10^{-10}$

Table 5.5: Pearson's correlation of course relatedness scores with human judgments.

To analyze the sensitivity of parameters α, β, and δ, the Pearson's correlation coef-

[10]The Cohen's kappa coefficient [14] of the data set is 0.35.

ficients are documented when the proposed approach is compared to human judgment. As Figure 5.6 shows, changing α, β, and δ do not have a huge impact on the result. The proposed approach maintains to be highly correlated with human judgment.

The proposed approach is more efficient than previous work. In the experiment, the average time needed to compare one pair of course descriptions ranged from 0.16 second (when enabling the caching of concept relatedness and information content) to 1 minute (without caching) on a 2.6Ghz Quad-Core PC. The most time-consuming part before comparing courses was to index all the Wikipedia tables in a MySQL database, which took overnight (same for ESA). It only took 15 minutes to go through 19K pages to build a hierarchy of depth $D = 4$. In contrast, ESA's first level semantic interpreter (which tokenizes every Wikipedia page to compute TF-IDF) took 7 days to build over the same 19K pages. Both implementations were single-threaded, coded in Python, and tested over the English Wikipedia of July 2011.

During the experiment, we have found some misclassified categories in the wikidump.[11] For example, "Category:Software" has over 350 subcategories with names similar to "Category:A-Class Britney Spears articles," or "Category:FA-Class Coca-Cola articles." None of these appears in the Wikipedia website or the Wikipedia API[12] as a subcategory of "Category:Software." More study is required on how they are formed.

[11]We have analyzed wikidumps of July 2011 and Oct 2010 and the problem persists in both versions.

[12]`https://www.mediawiki.org/wiki/API`

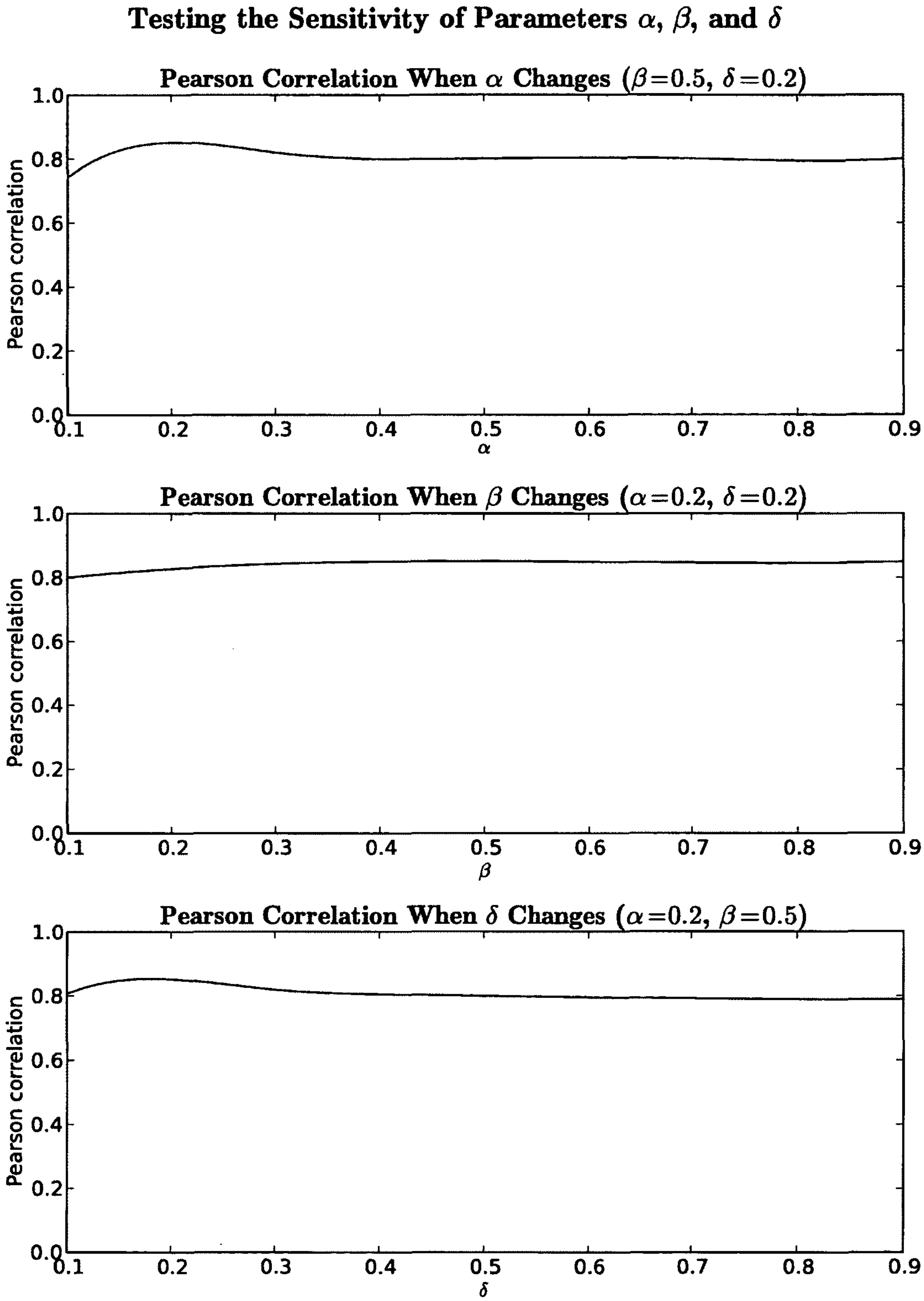


Figure 5.6: Pearson's correlation coefficients when α, β, or δ changes.

5.4 Walkthrough

This section shows how to compare two course descriptions using the proposed approach.

Given two course descriptions, the first step is to generate features for them. Each feature corresponds to a Wikipedia page.

5.4.1 Generate Features for Course C_1

> C_1: "[**Analysis of Algorithms**] Discusses basic methods for designing and analyzing efficient algorithms emphasizing methods used in practice. Topics include sorting, searching, dynamic programming, greedy algorithms, advanced data structures, graph algorithms (shortest path, spanning trees, tree traversals), matrix operations, string matching, NP completeness."

Input: "Analysis of Algorithms"

Given course title "Analysis of Algorithms" as input, removing its stop words returns "Analysis Algorithms." Its corresponding n-grams ($n \in [1, 3]$) are:

(1) Analysis Algorithms; (2) Analysis Algorithm; (3) Analysis; (4) Algorithms; (5) Analysis algorithms; (6) Analysis algorithm

In this step, both lower-case and upper-case of the first letter of each word (except the first word) are included. The first letter of each article title in Wikipedia is always capitalized therefore the lower-case form of such a letter is ignored.

After querying the Wikipedia redirection data with these n-grams, *unigram* "Algorithms" leads to Wikipedia page 775:"Algorithm".

Next, we feed the Wikipedia title data with these n-grams. Below shows the retrieved Wikipedia pages, displayed as page ID and page title pairs.

{7579257:"Analysis (journal)", 7043938:"Analysis (radio programme)", 15136106:"Analysis (disambiguation)", 1134:"Analysis"}

Using the disambiguation described in Algorithm 4 (page 66), the features for course title "Analysis of Algorithms" are: {1134:"Analysis", 775:"Algorithm"}.

Input: "Discusses basic methods for designing and analyzing efficient algorithms emphasizing methods used in practice. Topics include sorting, searching, dynamic programming, greedy algorithms, advanced data structures, graph algorithms (shortest path, spanning trees, tree traversals), matrix operations, string matching, NP completeness."

Similarly, given the course abstract as input, we generate its n-grams and disambiguate each of them. The features are:

{41985:"Shortest path problem", 597584:"Tree traversal", 455770:"Spanning tree", 18955875:"Tree", 1134:"Analysis", 18568:"List of algorithms", 56054:"Completeness", 775:"Algorithm", 144656:"Sorting", 8519:"Data structure", 93545:"Structure", 8560:"Design", 18985040:"Data"}

5.4.2 Generate Features for Course C_2

C_2: "[**Computing III**] Object-oriented programming. Classes, methods, polymorphism, inheritance. Object-oriented design. C++. UNIX. Ethical and social issues."

Input: "Computing III"

Generated feature: {5213:"Computing"}.

Input: "Object-oriented programming. Classes, methods, polymorphism, inheritance. Object-oriented design. C++. UNIX. Ethical and social issues."

Generated features:

{21347364:"Unix", 289862:"Social", 9258:"Ethics", 6111038:"Object-oriented design", 5311:"Computer programming", 72038:"C++", 27471338:"Object-oriented programming", 8560:"Design"}

Next, course title and course abstract pairs are measured separately for semantic relatedness.

5.4.3 Semantic Relatedness of Course Titles

The two feature vectors from the previous steps are given as input:

- {1134:"Analysis", 775:"Algorithm"}, and
- {5213:"Computing"}

The two vectors are joined into a unique list:

{1134:"Analysis", 775:"Algorithm", 5213:"Computing"}

This unique list is first compared with the first input vector and then compared with the second input vector. Each comparison implements Algorithm 5 to build the semantic vector. The semantic vectors of the two input vectors are:

- {(1134:"Analysis", 1134:"Analysis"): 1, (775:"Algorithm", 775:"Algorithm"): 1, (5213:"Computing", 775:"Algorithm"): 0}
- {(1134:"Analysis", 5213:"Computing"): 0, (775:"Algorithm", 5213:"Computing"): 0, (5213:"Computing", 5213:"Computing"): 1}

The two semantic vectors are then reweighted, taking into account of the information content of each term:

- {(1134:"Analysis", 1134:"Analysis"): 0.6407764239998172, (775:"Algorithm", 775:"Algorithm"): 0.6456048827818694, (5213:"Computing", 775:"Algorithm"): 0}

- {(1134:"Analysis", 5213:"Computing"): 0, (775:"Algorithm", 5213:"Computing"): 0, (5213:"Computing", 5213:"Computing"): 0.7442666866195237}

The cosine coefficient of the two semantic vectors is 0. Therefore the semantic relatedness of the two course titles is 0.

5.4.4 Semantic Relatedness of Course Abstracts

We perform the same steps on the two course abstracts. The two semantic vectors after the reweighing are:

- {(18955875:"Tree", 18955875:"Tree"): 0.6004465610898385, (8560:"Design", 8560:"Design"): 0.6372576231871345, (775:"Algorithm", 775:"Algorithm"): 0.6456048870338603, (9258:"Ethics", 18985040:"Analysis"): 0.17228798437234422, (18568:"List of algorithms", 18568:"List of algorithms"): 0.6615337484169812, (21347364:"Unix", 18985040:"List of algorithms"): 0, (41985:"Shortest path problem", 41985:"Shortest path problem"): 0.701933634947439, (93545:"Structure", 93545:"Structure"): 0.6657531978600502, (455770:"Spanning tree", 455770:"Spanning tree"): 0.7228965868728318, (18985040:"Data", 18985040:"Data"): 0.5891500959360039, (597584:"Tree traversal", 597584:"Tree traversal"): 0.66933432676929, (6111038:"Object-oriented design", 18985040:"List of algorithms"): 0, (72038:"C++", 18985040:"Tree traversal"): 0, (27471338:"Object-oriented programming", 18985040:"List of algorithms"): 0, (1134:"Analysis", 1134:"Analysis"): 0.6407764232932996, (56054:"Completeness", 56054:"Completeness"): 0.6034252467450718, (289862:"Social", 18985040:"Analysis"):

0, (8519:"Data structure", 8519:"Data structure"): 0.7115964608074017, (5311:"Computer programming", 18985040:"List of algorithms"): 0, (144656:"Sorting", 144656:"Sorting"): 0.6792343815182275}

- {(18955875:"Tree", 8560:"Unix"): 0, (8560:"Design", 8560:"Design"): 0.6372576331363273, (775:"Algorithm", 8560:"Object-oriented design"): 0, (9258:"Ethics", 9258:"Ethics"): 0.6418385539530528, (18568:"List of algorithms", 8560:"Object-oriented design"): 0, (21347364:"Unix", 21347364:"Unix"): 0.6823040208651913, (41985:"Shortest path problem", 8560:"Unix"): 0, (93545:"Structure", 8560:"Unix"): 0, (455770:"Spanning tree", 8560:"Object-oriented design"): 0, (18985040:"Data", 8560:"Unix"): 0, (597584:"Tree traversal", 8560:"C++"): 0, (6111038:"Object-oriented design", 6111038:"Object-oriented design"): 0.6373310613674149, (72038:"C++", 72038:"C++"): 0.5987680747856168, (27471338:"Object-oriented programming", 27471338:"Object-oriented programming"): 0.6241341564236399, (1134:"Analysis", 8560:"Ethics"): 0.17903591755327833, (56054:"Completeness", 8560:"Unix"): 0, (289862:"Social", 289862:"Social"): 0.510639379823403, (8519:"Data structure", 8560:"Object-oriented design"): 0, (5311:"Computer programming", 5311:"Computer programming"): 0.6236673881434073, (144656:"Sorting", 8560:"Object-oriented design"): 0}

The cosine coefficient of the two semantic vectors is 0.15.

Finally, by using Equation 5.7 (page 68) the semantic relatedness of the two course descriptions is 0.13.

5.5 Conclusion

This chapter presents a domain-specific algorithm to suggest equivalent courses based on analyzing their semantic relatedness using Wikipedia. Both accuracy and correlation suggest the proposed approach outperforms previous work. Future work includes the study of local knowledge (Ω), comparing our approach with ESA (Section 3.1.2.5), experimenting on more courses from more universities, and adapting our work to courses in other languages.

CHAPTER 6

SUMMARY AND FUTURE WORK

This dissertation addresses the problem of semantic relatedness by presenting an in-depth study of semantic relatedness measures in related work, the popular knowledge sources used by these measures, and the application of semantic relatedness on evaluation of course equivalencies. This study highlights the knowledge acquisition bottleneck, and further clarifies that Wikipedia as a knowledge source does not solve the knowledge acquisition bottleneck, unlike some previous work states.

To suggest course equivalencies, two approaches are proposed. The first approach is based on traditional knowledge sources such as WordNet and corpora. While this approach can measure courses from multiple domains and performs better than related work, due to the knowledge acquisition bottleneck in traditional knowledge sources, this approach is not promising on measurement of courses in technology-related domains that are heavily loaded with jargon.

Alternatively, the second approach uses Wikipedia as a knowledge source. Because of its openly-editable model, Wikipedia becomes the richest encyclopedia that is freely available and always up-to-date. In recent years, there has been an increasing interest in using Wikipedia to tackle various problems. Unfortunately, the exponential growth of Wikipedia is often neglected in related work. As a result, most semantic relatedness measures using Wikipedia in the related work are highly inefficient and they are becoming less and less efficient as the size of Wikipedia increases. To address the problem, the second approach proposes a domain-specific semantic relatedness measure based on part of Wikipedia that analyzes course descriptions to suggest whether a course can be transferred from one institution to another. It is shown that while the second approach removes over 71% of Wikipedia articles to maintain its high efficiency, it still performs better than related work and reaches a high correlation compared to human judgment.

Institutions who opt to make their course descriptions freely available online often publish their data in arbituary formats. Additionally, the course equivalencies listed in some transfer dictionaries are sparse and out of date. Because of these issues, it is very difficult to gather a large data set of equivalent and nonequivalent course descriptions. The data sets used in this study were acquired by scraping course descriptions off different websites. It would be interesting to use our approaches on a larger data set including more universities.

In the future we would like to explore how to utilize parameter Ω (Equation 5.7, page 68) to incorporate local knowledge including known course equivalencies and user feedback.

Our approaches only focus on course titles and course abstracts. Future work can bring in more parameters to tailor to the different needs of various institutions. These parameters may include the level of a course, the number of times a class meets, and the textbook being used. Another direction is to take advantage of multilingual nature of Wikipedia and apply our second approach to other languages.

APPENDIX A

PENN TREEBANK PART OF SPEECH TAGS

Originally provided by the Penn Treebank Project[1] to annotate text with *part of speech* (POS) tags, the Penn Treebank POS tags are widely used in related work for POS tagging.

Table A.1: Penn Treebank POS Tags

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential there
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass

Continued on next page

[1] `http://www.cis.upenn.edu/~treebank/`

Table A.1 – continued from previous page

Number	Tag	Description
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

APPENDIX B
STOP WORDS

The stop words in Chapter 5 include the 127 stop words from the Snowball English stop word list [54] and 129 domain stop words. These words are listed below.

a
about
above
after
again
against
all
also
am
an
and
any
are
area
areas
as
aspects
association
associations
at
award
awards
basic
be
because
been
before
being
below
between
book
books
both
business
but
by
can
category:beam
category:packaging
category:projects
companies
company
complete
concept
concepts
conference
conferences
countries
country
course
courses
coursework
courseworks
cover
covered
covers
describe
description
did
do
documentaries
documentary
does
doing
don
down
during
each
eight
emphasis
emphasize
emphasizes

end
ends
events
example
examples
exercise
exercises
experience
experiences
faculty
few
fifth
first
five
focus
focuses
for
four
fourth
from
further
graduation
had
has
have
having
he
her
here
hers
herself
him
himself
his
hours
how
i
if
ii
iii
in
include
includes
including
into
introduce
introduces
introduction
introductions
is
issue
issues
it
its
itself
iv
just
lab
learn
learns
lecture
lectures
man
may
me
men
mentor
mentors
method
methods
more
most
my
myself
nine
no
nor
not
now
of
off
on
once
one
only
or
organization
organizations
other
our
ours
ourselves
out
over
own
people
person
reading
readings
require
requirement
requires
s
same
second
see
sees
serve

serves
seven
she
should
simple
six
skill
skills
so
software
solve
solves
solving
some
student
students
studies
study
such
t
ten
than
that
the
their
theirs
them
themselves
then
there
these
they
third
this
those
three
through
time
to
too
topic
topics
two
under
universities
university
until
up
use
uses
using
various
very
via
was
we
well
were
what
when
where
which
while
who
whom
why
will
with
within
work
works
you
your
yours
yourself
yourselves

APPENDIX C

HUMAN JUDGMENT DATASET OF COMPUTER SCIENCE COURSE EQUIVALENCIES

Table C.1 reports the human judgment data set on Compute Science (CS) course equivalencies [68], based on the evaluations of 6 annotators consisting of CS students and professors. To create this data set, each of the 6 annotators was given a list of 32 pairs of CS courses, with only course titles and descriptions. They independently evaluated whether each pair is equivalent on a scale from 1 to 5. Next, the mean value of their evaluations for each pair was calculated, and the scale was converted from [1,5] to [0,1].

An electronic copy of the data set can be obtained from `http://github.com/beibeiyang/semcourse`. The data set is released under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.[1]

Table C.1: Human Judgment Dataset of Computer Science Course Equivalencies

No.	Course A	Course B	Score
1.	**Computer Science I** First course in Computer Science. Introduces the fundamental concepts of computer programming with an object-oriented language with an emphasis on analysis and design. Topics include data types, selection and iteration, instance variables and methods, arrays, files, and the mechanics of running, testing and debugging.	**Undeclared Science Seminar** Discussions will be conducted on a wide range of topics in the sciences to familiarize the student with the programs, procedures, research, and educational opportunities at the University.	0.24

Continued on next page

[1] `http://creativecommons.org/licenses/by-nc-sa/3.0/`

Table C.1 – continued from previous page

No	Course A	Course B	Score
2.	**Programming I** This foundational course for computer science majors introduces the fundamental concepts of programming from an object-centric perspective using Java. Includes a brief introduction to computing (historical development, computing systems, algorithms, and the nature of programming languages) and the object-oriented paradigm for software development. Topics include: objects, classes, methods, simple data types, control structures, and the use of indexed-list data structures such as arrays or strings. Includes discussion of the ethics and responsibility of computer professionals with respect to information rights.	**Exploring the Internet** This course focuses on the primary tools used to navigate the Internet from a Windows desktop: e-mail and the web browsers. In addition, this course covers many of the other applications of the Internet: ftp, listserve, newsgroups, chat, search engines, and portals. Students will complete hands-on exercises, including construction of their personal web page. Not for computer science majors.	0.20
3.	**Intermediate Programming Using C++** This course is the second course in the software development sequence. It continues the idea of using programming and its constructs to solve problems. The student's understanding of variables, arrays, if, if else, loops, and functions will be reinforced, while introducing the student to the object oriented C++ programming language. Additionally the student will be introduced to pointers and structures, and selected preprocessor directives as well as bit manipulations.	**Media Computing** Introduction to computer programming using multimedia applications. Programming data structures are covered by manipulating pictures, sounds and video. Linear Data structures such as arrays and matrices are manipulated in a computer programming language Java and C.	0.52
4.	**Computer Science I** First course in Computer Science. Introduces the fundamental concepts of computer programming with an object-oriented language with an emphasis on analysis and design. Topics include data types, selection and iteration, instance variables and methods, arrays, files, and the mechanics of running, testing and debugging.	**Operating Systems** Presents an introduction to major operating systems and their components. Topics include processes, concurrency and synchronization, deadlock, processor allocation, memory management, I/O devices and file management, and distributed processing. Techniques in operating system design, implementation, and evaluation will be examined.	0.36

Continued on next page

Table C.1 – continued from previous page

No	Course A	Course B	Score
5.	**Computer Science II** A continuation of CIS141 Computer Science I emphasizing the development of data structures to organize information in solving problems with computers. Typical structures include arrays, stacks, queues, linked lists, and trees. Coverage will include searching, sorting and algorithm analysis. Laboratory projects will give students the opportunity to implement these data structures.	**Computing III** Object-oriented programming. Classes, methods, polymorphism, inheritance. Object-oriented design. C++. UNIX. Ethical and social issues.	0.32
6.	**Intermediate Programming Using C++** This course is the second course in the software development sequence. It continues the idea of using programming and its constructs to solve problems. The student's understanding of variables, arrays, if, if else, loops, and functions will be reinforced, while introducing the student to the object oriented C++ programming language. Additionally the student will be introduced to pointers and structures, and selected preprocessor directives as well as bit manipulations.	**Computing IV** Development of large software projects. Software engineering principles and practice. Object-oriented analysis and design. CASE productivity aids. Development techniques for program-translation software and web software.	0.36
7.	**Computer Science I** First course in Computer Science. Introduces the fundamental concepts of computer programming with an object-oriented language with an emphasis on analysis and design. Topics include data types, selection and iteration, instance variables and methods, arrays, files, and the mechanics of running, testing and debugging.	**Honors Project I** This course provides an undergraduate research experience for Computer Science majors enrolled in the Honors Program. Each student develops a project idea in consultation with the instructor. The student writes a proposal for the project, reads the relevant literature, performs the project, writes a project report or thesis, and makes an oral presentation about the project.	0.20

Continued on next page

Table C.1 – continued from previous page

No	Course A	Course B	Score
8.	**Programming III** This course emphasizes advanced programming techniques in Java, an object-oriented programming language. Students will produce console and GUI applications that interact with files and streams. Advanced programming concepts such as exception handling, multithreading, layout managers, image animation, and audio will also be covered.	**Tangible Interaction Design** Tangible Interaction Design focuses on understanding how people interact with the designed things in the everyday world around us. The course is project-oriented with two significant projects and a series of smaller lab assignments. Through these assignments, students will learn elements of graphical communication and principles of interaction in computationally-enabled devices.	0.36
9.	**Analysis of Algorithms** Description: Discusses basic methods for designing and analyzing efficient algorithms emphasizing methods used in practice. Topics include sorting, searching, dynamic programming, greedy algorithms, advanced data structures, graph algorithms (shortest path, spanning trees, tree traversals), matrix operations, string matching, NP completeness.	**Undeclared Science Seminar** Discussions will be conducted on a wide range of topics in the sciences to familiarize the student with the programs, procedures, research, and educational opportunities at the University.	0.20
10.	**Computer Programming Concepts** This course introduces students to the ideas that make computers work and to the concepts underlying object-oriented programming languages such as ActionScript, Java or C++. In the first part of the course, students will learn about binary numbers, the logic structures within the computer, and the basic computer programming constructs. Students will see examples of how programming constructs are implemented in a variety of programming languages. In the second part of the course, students will develop their own computer programs in a widely-used object-oriented language in the web design and interactive media industries such as ActionScript, Java or C++. The course format combines lecture and hands-on lab.	**Graphical User Interface Programming I** This is a first course in the design and implementation of graphical user interfaces (GUIs) for windowing environments. The course involves numerous programming projects that are evaluated on design and layout of the user interface, coding style, and comprehensiveness of documentation. The course may be taken on its own, but is intended to be followed by 91.462 to complete a two-course CS project sequence.	0.44

Continued on next page

Table C.1 – continued from previous page

No	Course A	Course B	Score
11.	**Computer Programming Concepts** This course introduces students to the ideas that make computers work and to the concepts underlying object-oriented programming languages such as ActionScript, Java or C++. In the first part of the course, students will learn about binary numbers, the logic structures within the computer, and the basic computer programming constructs. Students will see examples of how programming constructs are implemented in a variety of programming languages. In the second part of the course, students will develop their own computer programs in a widely-used object-oriented language in the web design and interactive media industries such as ActionScript, Java or C++. The course format combines lecture and hands-on lab.	**Media Computing** Introduction to computer programming using multimedia applications. Programming data structures are covered by manipulating pictures, sounds and video. Linear Data structures such as arrays and matrices are manipulated in a computer programming language Java and C.	0.76
12.	**Analysis of Algorithms** Description: Discusses basic methods for designing and analyzing efficient algorithms emphasizing methods used in practice. Topics include sorting, searching, dynamic programming, greedy algorithms, advanced data structures, graph algorithms (shortest path, spanning trees, tree traversals), matrix operations, string matching, NP completeness.	**Analysis of Algorithms** Development of more sophisticated ideas in data type and structure, with an introduction to the connection between data structures and the algorithms they support. Data abstraction. Controlled access structures. Trees, lists, graphs, arrays; algorithms design strategies; backtracking, greedy storage, divide and conquer, branch and bound. Elementary techniques for analysis; recursion equations, estimations methods, elementary combinatorial arguments. Examination of problem areas such as searching, sorting, shortest path, matrix and polynomial operations, and the indicated representations and algorithms. The student will use the techniques learned in this course and in previous courses to solve a number of logically complex programming problems.	0.92

Continued on next page

Table C.1 – continued from previous page

No	Course A	Course B	Score
13.	**Analysis of Algorithms** Description: Discusses basic methods for designing and analyzing efficient algorithms emphasizing methods used in practice. Topics include sorting, searching, dynamic programming, greedy algorithms, advanced data structures, graph algorithms (shortest path, spanning trees, tree traversals), matrix operations, string matching, NP completeness.	**Assembly Language Programming** Presents the organization and operation of a conventional computer, including principal instruction types, data representation, addressing modes, program control, I/O, assembly language programming, including instruction mnemonics, symbolic addresses, assembler directives, system calls, and macros, the usage of text editors, symbolic debuggers, and loaders, and the use of pseudocode in guiding structured assembly language programming.	0.28
14.	**Introduction to Programming** This is a first course of a three course sequence in C++ programming for the student with little or no programming experience. The course introduces students to problem-solving methods, algorithm development, and implementing program code in C++. Topics covered will include procedural and data abstractions, program design, debugging, testing, and documentation. The course will also include both built-in and programmer defined data types, control structures, library functions, programmer defined functions with parameter passing, arrays, structures, as well as an introduction to object oriented programming using classes. Laboratory exercise will be implemented using the C++ programming language.	**Computing I** Introduction to computing environments: introduction to an integrated development environment; C, C++, or a similar language. Linear data structures; arrays, records, and linked lists. Abstract data types, stacks, and queues. Simple sorting via exchange, selection, and insertion, Basic file I/O. Programming style documentation and testing. Ethical and social issues.	0.92

Continued on next page

Table C.1 – continued from previous page

No	Course A	Course B	Score
15.	**Introduction to Programming** This is a first course of a three course sequence in C++ programming for the student with little or no programming experience. The course introduces students to problem-solving methods, algorithm development, and implementing program code in C++. Topics covered will include procedural and data abstractions, program design, debugging, testing, and documentation. The course will also include both built-in and programmer defined data types, control structures, library functions, programmer defined functions with parameter passing, arrays, structures, as well as an introduction to object oriented programming using classes. Laboratory exercise will be implemented using the C++ programming language.	**Artificial Intelligence** Discusses LISP, tree and graph searching algorithms: breadth first, depth first, and uniform cost. Also covers heuristic search methods, admissibility, and games: mini-max, alphaBeta. Students will learn theorem proving and question answering.	0.20
16.	**Introduction to Programming** Provides an introduction to computer programming (software) concepts and functions. Introduces problem-solving methods and algorithm development using software programming. Includes procedural and data abstractions, program design, debugging, testing, and documentation. Covers data types, control structures, functions, parameter passing, library functions, and arrays. Laboratory exercises in C++.	**Computer Security** Basic concepts of cryptography, data security, information theory, complexity, number theory, and finite field theory; encryption algorithms including the Data Encryption Standard (DES) and public key systems; incorporating cryptographic controls into computers; key management; access controls; information flow controls; and inference controls.	0.24
17.	**Introduction to Programming** Provides an introduction to computer programming (software) concepts and functions. Introduces problem-solving methods and algorithm development using software programming. Includes procedural and data abstractions, program design, debugging, testing, and documentation. Covers data types, control structures, functions, parameter passing, library functions, and arrays. Laboratory exercises in C++.	**Computing I** Introduction to computing environments: introduction to an integrated development environment; C, C++, or a similar language. Linear data structures; arrays, records, and linked lists. Abstract data types, stacks, and queues. Simple sorting via exchange, selection, and insertion, Basic file I/O. Programming style documentation and testing. Ethical and social issues.	0.92

Continued on next page

Table C.1 – continued from previous page

No	Course A	Course B	Score
18.	**Programming II** This programming course emphasizes object-oriented design. Topics include class construction, data abstraction, inheritance, overloading, overriding, exceptions, encapsulation, static classes and polymorphism. Students use an Integrated Development Environment (IDE) to create applications in Java.	**Computing I** Introduction to computing environments: introduction to an integrated development environment; C, C++, or a similar language. Linear data structures; arrays, records, and linked lists. Abstract data types, stacks, and queues. Simple sorting via exchange, selection, and insertion, Basic file I/O. Programming style documentation and testing. Ethical and social issues.	0.40
19.	**Programming II** This programming course emphasizes object-oriented design. Topics include class construction, data abstraction, inheritance, overloading, overriding, exceptions, encapsulation, static classes and polymorphism. Students use an Integrated Development Environment (IDE) to create applications in Java.	**Robotics I** An introduction to robotics, including laboratory. In the lab, students build and program robots. Topics to be covered include sensors, locomotion, deliberative architectures, reactive architectures, and hybrid architectures.	0.20
20.	**Data Structure and Algorithms I** Students are individually responsible for the formal specification, design, implementation and proof of correctness of the abstract data type sets, bags, functions, sequences, stacks, queues, and strings. Special emphasis will be given to searching and sorting algorithms.	**Analysis of Algorithms** Development of more sophisticated ideas in data type and structure, with an introduction to the connection between data structures and the algorithms they support. Data abstraction. Controlled access structures. Trees, lists, graphs, arrays; algorithms design strategies; backtracking, greedy storage, divide and conquer, branch and bound. Elementary techniques for analysis; recursion equations, estimations methods, elementary combinatorial arguments. Examination of problem areas such as searching, sorting, shortest path, matrix and polynomial operations, and the indicated representations and algorithms. The student will use the techniques learned in this course and in previous courses to solve a number of logically complex programming problems.	0.64

Continued on next page

Table C.1 – continued from previous page

No	Course A	Course B	Score
21.	**Data Structure and Algorithms I** Students are individually responsible for the formal specification, design, implementation and proof of correctness of the abstract data type sets, bags, functions, sequences, stacks, queues, and strings. Special emphasis will be given to searching and sorting algorithms.	**Computing I** Introduction to computing environments: introduction to an integrated development environment; C, C++, or a similar language. Linear data structures; arrays, records, and linked lists. Abstract data types, stacks, and queues. Simple sorting via exchange, selection, and insertion, Basic file I/O. Programming style documentation and testing. Ethical and social issues.	0.28
22.	**Data Structure and Algorithms I** Students are individually responsible for the formal specification, design, implementation and proof of correctness of the abstract data type sets, bags, functions, sequences, stacks, queues, and strings. Special emphasis will be given to searching and sorting algorithms.	**Media Computing** Introduction to computer programming using multimedia applications. Programming data structures are covered by manipulating pictures, sounds and video. Linear Data structures such as arrays and matrices are manipulated in a computer programming language Java and C.	0.44
23.	**Data Structure and Algorithms I** Students are individually responsible for the formal specification, design, implementation and proof of correctness of the abstract data type sets, bags, functions, sequences, stacks, queues, and strings. Special emphasis will be given to searching and sorting algorithms.	**Graphical User Interface Programming I** This is a first course in the design and implementation of graphical user interfaces (GUIs) for windowing environments. The course involves numerous programming projects that are evaluated on design and layout of the user interface, coding style, and comprehensiveness of documentation. The course may be taken on its own, but is intended to be followed by 91.462 to complete a two-course CS project sequence.	0.28
24.	**Data Structures** Introduction to data structures and algorithms. Topics include lists, stacks, queues, trees, heaps, graphs, and sorting and searching algorithms including hash coding.	**Computing II** Pointers. Lists, stacks and queues. Binary trees, AVL trees, n-ary trees. Advanced sorting via quicksort, heapsort, etc. Characters and strings. Graphs. Advanced file techniques. Recursion. Programming style, documentation, and testing. Ethical and social issues This course includes extensive laboratory work.	0.80
		Continued on next page	

Table C.1 – continued from previous page

No	Course A	Course B	Score
25.	**Data Structures** Introduction to data structures and algorithms. Topics include lists, stacks, queues, trees, heaps, graphs, and sorting and searching algorithms including hash coding.	**Analysis of Algorithms** Development of more sophisticated ideas in data type and structure, with an introduction to the connection between data structures and the algorithms they support. Data abstraction. Controlled access structures. Trees, lists, graphs, arrays; algorithms design strategies; backtracking, greedy storage, divide and conquer, branch and bound. Elementary techniques for analysis; recursion equations, estimations methods, elementary combinatorial arguments. Examination of problem areas such as searching, sorting, shortest path, matrix and polynomial operations, and the indicated representations and algorithms. The student will use the techniques learned in this course and in previous courses to solve a number of logically complex programming problems.	0.56
26.	**Data Structures** Introduction to data structures and algorithms. Topics include lists, stacks, queues, trees, heaps, graphs, and sorting and searching algorithms including hash coding.	**Data Communications I** This course provides an introduction to fundamental concepts in the design and implementation of computer communication networks, their protocols, and applications. Topics include: TCP/IP and OSI layered network architectures and associated protocols, application layer, network programming API (sockets), transport, congestion, flow control, routing, addressing, autonomous systems, multicast and link layer. Examples will be drawn primarily from the Internet.	0.32

Continued on next page

Table C.1 – continued from previous page

No	Course A	Course B	Score
27.	**Computer Organization/Assembly Language** Introduction to binary, octal and hexadecimal number systems, machine language and machine architecture. Assembly language topics include the assembly process, arithmetic, addressing modes, subprograms, procedures, input/output and conditional assembly.	**Assembly Language Programming** Presents the organization and operation of a conventional computer, including principal instruction types, data representation, addressing modes, program control, I/O, assembly language programming, including instruction mnemonics, symbolic addresses, assembler directives, system calls, and macros, the usage of text editors, symbolic debuggers, and loaders, and the use of pseudocode in guiding structured assembly language programming.	0.96
28.	**Computer Organization/Assembly Language** Introduction to binary, octal and hexadecimal number systems, machine language and machine architecture. Assembly language topics include the assembly process, arithmetic, addressing modes, subprograms, procedures, input/output and conditional assembly.	**Organization of Programming Languages** Analytical approach to the study of programming languages. Description of the salient features of the imperative, functional, logical, and object-oriented programming paradigms in a suitable metalanguage such as Scheme. Topics include iteration, recursion, higher-order functions, types, inheritance, unification, message passing, orders of evaluation, and scope rules. Elementary syntactic and semantic descriptions. Implementation of simple interpreters.	0.40
29.	**Computer Organization/Assembly Language** Introduction to binary, octal and hexadecimal number systems, machine language and machine architecture. Assembly language topics include the assembly process, arithmetic, addressing modes, subprograms, procedures, input/output and conditional assembly.	**Data Mining** This introductory data mining course will give an overview of the models and algorithms used in data mining, including association rules, classification, clustering, etc. The course will teach the theory of these algorithms and students will learn how and why the algorithms work through computer labs.	0.24

Continued on next page

Table C.1 – continued from previous page

No	Course A	Course B	Score
30.	**Algorithms and Data** Introduces the basic principles and techniques for the design, analysis, and implementation of efficient algorithms and data representations. Discusses asymptotic analysis and formal methods for establishing the correctness of algorithms. Considers divide-and-conquer algorithms, graph traversal algorithms, and optimization techniques. Introduces information theory and covers the fundamental structures for representing data. Examines flat and hierarchical representations, dynamic data representations, and data compression. Concludes with a discussion of the relationship of the topics in this course to complexity theory and the notion of the hardness of problems.	**Analysis of Algorithms** Development of more sophisticated ideas in data type and structure, with an introduction to the connection between data structures and the algorithms they support. Data abstraction. Controlled access structures. Trees, lists, graphs, arrays; algorithms design strategies; backtracking, greedy storage, divide and conquer, branch and bound. Elementary techniques for analysis; recursion equations, estimations methods, elementary combinatorial arguments. Examination of problem areas such as searching, sorting, shortest path, matrix and polynomial operations, and the indicated representations and algorithms. The student will use the techniques learned in this course and in previous courses to solve a number of logically complex programming problems.	0.92
31.	**Algorithms and Data** Introduces the basic principles and techniques for the design, analysis, and implementation of efficient algorithms and data representations. Discusses asymptotic analysis and formal methods for establishing the correctness of algorithms. Considers divide-and-conquer algorithms, graph traversal algorithms, and optimization techniques. Introduces information theory and covers the fundamental structures for representing data. Examines flat and hierarchical representations, dynamic data representations, and data compression. Concludes with a discussion of the relationship of the topics in this course to complexity theory and the notion of the hardness of problems.	**Compiler Construction I** Includes both theory and practice. A study of grammars; specification and classes; the translation pipeline: lexical analysis, parsing, semantic analysis, code generation and optimization; and syntax-directed translation. Use of automatic generation tools in the actual production of a complete compiler for some language.	0.24

Continued on next page

Table C.1 – continued from previous page

No	Course A	Course B	Score
32.	**Algorithms and Data** Introduces the basic principles and techniques for the design, analysis, and implementation of efficient algorithms and data representations. Discusses asymptotic analysis and formal methods for establishing the correctness of algorithms. Considers divide-and-conquer algorithms, graph traversal algorithms, and optimization techniques. Introduces information theory and covers the fundamental structures for representing data. Examines flat and hierarchical representations, dynamic data representations, and data compression. Concludes with a discussion of the relationship of the topics in this course to complexity theory and the notion of the hardness of problems.	**Software Project I** Specification, design, and implementation of a one- or two-semester software project proposed to a directing faculty member. Projects may be proposed as a one- or two-semester effort based on faculty approval. A two-semester effort requires subsequent registration for 91.402. Prerequisite: Students must submit a proposal to the directing faculty member, obtain his/her signed approval, and forward a copy of the signed proposal to department chairperson	0.24

GLOSSARY

antonymy

An antonym is a word that expresses a meaning opposed to the meaning of another word. For example, "fast" is an antonym of "slow."

bag-of-words model

A bag-of-words model treats a text passage as an unordered collection of words and ignores other information such as word orders and grammar.

big data

Big data refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze.

concept

A concept corresponds to one meaning of a word. A word may have multiple concepts. A concept typically corresponds to a node in the WordNet hierarchy.

corpus

A corpus is a large and structured set of texts that may or may not be annotated.

distributional hypothesis

Distributional hypothesis is the theory that words that occur in the similar contexts tend to have similar meanings.

expert system

An expert system is a computer system that emulates the decision-making ability of a human expert.

holonymy

A *holonym* is a word that names the whole of which a given word is a part. For example, "computer" is a holonym for "CPU" and "memory."

hypernymy

A *hypernym* is a word that is more generic than a given word. For example, "cutlery" is a hypernym of "knife","fork", and "spoon."

hyponymy

A *hyponym* is a a word that is more specific than a given word. For example, "knife","fork", and "spoon" are hyponyms of "cutlery."

information content
: Information content is a metric to denote the importance of a word in a corpus. A word is given a higher information content (IC) value if it's more important. For example, "computer" generally has a higher IC value than "and" in an English corpus.

IS_A network
: IS_A networks are broadly used in areas such as artificial intelligence, database, and software engineering for knowledge representation and software design. If concept A is logically a subclass of concept B, we say that A and B have an is-a link. An is-a network is a hierarchical structure of a collection of these is-a links.

knowledge acquisition
: Knowledge acquisition is the transfer and transformation of problem-solving expertise from some knowledge source to a program.

knowledge acquisition bottleneck
: Knowledge acquisition bottleneck is a common problem that occurs in the knowledge acquisition process in expert systems.

lexicon
: Lexicon represents words and phrases that can be used in the text. The lexicon of a language is its vocabulary.

lowest common ancestor
: Sometimes called *Least Common Subsumer*, the lowest common ancestor of two nodes m and n in a rooted tree is defined as the lowest node in the tree that has both m and n as descendants.

malapropism
: Malapropism is the usually unintentionally humorous misuse or distortion of a word or phrase.

meronymy
: A *meronym* is a word that names a part of a larger whole. For example, "CPU" and "memory" are meronyms of "computer."

n-gram
: An n-gram is a contiguous sequence of n items from a given sequence of text or speech.

natural language processing
: Natural language processing is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and natural languages.

ontology

An ontology is a formal specification of a shared conceptualization [24]. In other words, an ontology is a description of the concepts and relationships that exist for an agent or a community of agents.

parsed corpus

Sometimes called a treebank, a parsed corpus is a corpus that is pre-processed and annotated with metadata.

part of speech

A part of speech is a category to which a word is assigned in accordance with its syntactic functions.

polysemous

A polysemous word has multiple senses (meanings).

sense

A sense corresponds to one meaning of a word. A word may have multiple senses.

singular value decomposition

If A is a $m \times n$ real matrix with $m > n$, then A can be written using the singular value decomposition of the form: $A = U \cdot D \cdot V^T$, where U is an $m \times m$ matrix, D is an $m \times n$ matrix, and V^T is an $n \times n$ matrix. U and V have orthogonal columns so that $U^T \cdot U = 1$, and $V^T \cdot V = 1$. Besides real matrices, singular value decomposition can also be applied to complex matrices.

stemming

The goal of stemming is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

stop word

Stop words are words that are filtered out prior to, or after, processing of natural language data. Any group of words can be chosen as the stop words for a given purpose. General stop words include some of the most common words such as "a," "the," "of," and "in."

synonym

A synonym is a word that means the same as another word, such as bucket and pail.

synset

A synset is a synonyms set in WordNet; a set of words that are interchangeable in some context without changing the true value of the preposition in which they are embedded.

TF-IDF
: TF-IDF stands for Term Frequency–Inverse Document Frequency, a weighting scheme often used in information retrieval and text mining.

treebank
: A treebank is an annotated corpus.

UML
: UML refers to the University of Massachusetts Lowell in this study.

unigram
: A unigram is a n-gram with $n = 1$.

word sense disambiguation
: Word sense disambiguation is the process of distinguishing the correct sense of a polysemous word.

BIBLIOGRAPHY

[1] Abhishek, Vibhanshu, and Hosanagar, Kartik. Keyword generation for search engine advertising using semantic similarity between terms. In *Proceedings of the 9th International Conference on Electronic Commerce* (New York, NY, USA, 2007), ACM, pp. 89–94.

[2] Bernard, John R. L. *The Macquarie Thesaurus.* Macquarie Library, Sydney, Australia, 1984.

[3] Berners-Lee, Tim, Hendler, James, and Lassila, Ora. The semantic web. *Scientific American 284*, 5 (2001), 34–43.

[4] Bird, Steven, Klein, Ewan, and Loper, Edward. *Natural Language Processing with Python.* O'Reilly, 2009.

[5] Bollegala, Danushka, Matsuo, Yutaka, and Ishizuka, Mitsuru. Measuring semantic similarity between words using web search engines. In *Proceedings of the 16th International Conference on World Wide Web* (New York, NY, USA, 2007), ACM, pp. 757–766.

[6] Bounova, Gergana. *Topological Evolution of Networks: Case Studies in the US Airlines and Language Wikipedias.* PhD thesis, MIT, 2011.

[7] Brachman, R.J. What is-a is and isn't: An analysis of taxonomic links in semantic networks. *Computer 16* (1983), 30–36.

[8] Buckley, Chris, Salton, Gerard, Allan, James, and Singhal, Amit. Automatic query expansion using smart: Trec 3. In *TREC* (1994), pp. 0–.

[9] Budanitsky, Alexander, and Hirst, Graeme. Evaluating Wordnet-based measures of lexical semantic relatedness. *Computational Linguistics 32* (2006), 13–47.

[10] Burgess, Curt, Livesay, Kay, and Lund, Kevin. Explorations in context space: words, sentences, discourse. *Discourse Processes 25* (1998), 211–257.

[11] Burnard, Lou. User reference guide for the british national corpus. *Technical report* (2000).

[12] Buscaldi, David, and Rosso, Paolo. Mining knowledge from Wikipedia from the question answering task. In *Proceedings of the 5th International Conference on Language Resources & Evaluation* (Genoa, Italy, 2006).

[13] Cilibrasi, Rudi L., and Vitanyi, Paul M. B. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering 19* (2007), 370–383.

[14] Cohen, Jacob. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin 70*, 4 (October 1968), 213–220.

[15] Collins, Allan M., and Quillian, M. Ross. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior 8*, 2 (April 1969), 240–247.

[16] Collins-Thompson, Kevyn, and Callan, Jamie. Query expansion using random walk models. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management* (New York, NY, USA, 2005), ACM, pp. 704–711.

[17] Curran, James R. *From Distributional to Semantic Similarity.* PhD thesis, University of Edinburgh, 2004.

[18] Fellbaum, Christiane, Ed. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication).* The MIT Press, May 1998.

[19] Firth, John R. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis* (1957), 1–32.

[20] Francis, W.Ñelson, and Kučera, Henry. *Brown Corpus Manual.* Technical report, Brown University, Providence, 1979.

[21] Gabrilovich, Evgeniy, and Markovitch, Shaul. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on AI* (2007).

[22] Gabrilovich, Evgeniy, and Markovitch, Shaul. Wikipedia-based semantic interpretation for NLP. *Journal of Artificial Intelligence Research 34* (2009), 443–498.

[23] Gentle, James E. *Numerical Linear Algebra for Applications in Statistics.* Springer, 1998.

[24] Gruber, Thomas R. A translation approach to portable ontology specifications. *Knowledge Acquisition 5*, 2 (1993), 199–220.

[25] Harris, Zellig S. *Mathematical structures of language*, vol. no. 21. Interscience Publishers, New York, 1968.

[26] Hayes-Roth, Frederick, Waterman, Donald A., and Lenat, Douglas B. *Building expert systems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983.

[27] Hirst, Graeme, and St-Onge, David. *WordNet: An electronic lexical database.* The MIT Press, Cambridge, MA, 1998, ch. Lexical chains as representations of context for the detection and correction of malapropisms, pp. 305–332.

[28] Ide, N. and Macleod, C.'The American national corpus: A standardized resource of American english. *In Proceedings of Corpus Linguistics 2001* (2001).

[29] Jiang, Jay J., and Conrath, David W. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference Research in Computational Linguistics* (1997), pp. 19–33.

[30] Joachims, Thorsten. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Internation Conference on Machine Learning* (1997).

[31] Kozima, Hideki, and Furugori, Teiji. Similarity between words computed by spreading activation on an english dictionary. In *Proceedings of the 6th conference on European chapter of the Association for Computational Linguistics* (Stroudsburg, PA, USA, 1993), EACL '93, Association for Computational Linguistics, pp. 232–239.

[32] Landauer, Thomas K, Foltz, Peter W., and Laham, Darrell. An introduction to latent semantic analysis. *Discourse Processes 25*, 2-3 (1998), 259–284.

[33] Leacock, Claudia, and Chodorow, Martin. *Combining local context and WordNet similarity for word sense identification.* The MIT Press, Cambridge, MA, 1998, pp. 265–283.

[34] Lenat, Douglas B. Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM 38*, 11 (November 1995), 33–38.

[35] Lesk, Michael. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation* (New York, NY, USA, 1986), SIGDOC '86, ACM, pp. 24–26.

[36] Li, Yuhua, Bandar, Zuhair A., and McLean, David. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering* (2003), 871–882.

[37] Li, Yuhua, McLean, David, Bandar, Zuhair A., O'Shea, James D., and Crockett, Keeley. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering 18*, 8 (2006), 1138–1150.

[38] Lin, Dekang. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics* (Stroudsburg, PA, USA, 1997), ACL '98, Association for Computational Linguistics, pp. 64–71.

[39] Lin, Dekang. An information-theoretic definition of similarity. In *ICML* (1998), Jude W. Shavlik, Ed., Morgan Kaufmann, pp. 296–304.

[40] Manning, Christopher D., Raghavan, Prabhakar, and Schütze, Hinrich. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[41] McCarthy, Diana, Koeling, Rob, Weeds, Julie, and Carroll, John. Using automatically acquired predominant senses for word sense disambiguation. In *Proceedings of the ACL SENSEVAL-3 Workshop* (2004), pp. 151–154.

[42] McKinsey Global Institute. Big data: The next frontier for innovation, competition, and productivity. Tech. rep., May 2011.

[43] Mihalcea, Rada, and Csomai, Andras. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the 16th ACM Conference on Information & Knowledge Management* (2007), pp. 233–242.

[44] Miller, George A., Leacock, Claudia, Tengi, Randee, and Bunker, Ross T. A semantic concordance. In *Proceedings of the workshop on Human Language Technology* (Stroudsburg, PA, USA, 1993), HLT '93, Association for Computational Linguistics, pp. 303–308.

[45] Mohammad, Saif. *Measuring Semantic Distance Using Distributional Profiles of Concepts.* PhD thesis, University of Toronto, Toronto, Canada, 2008.

[46] Mohammad, Saif, and Hirst, Graeme. Distributional measures as proxies for semantic distance: A survey. *Computational Linguistics 1*, 1 (2006).

[47] Mohammad, Saif, and Hirst, Graeme. Distributional measures of concept-distance: A task-oriented evaluation. Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing.

[48] Morris, Jane, and Hirst, Graeme. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics 17*, 1 (March 1991), 21–48.

[49] National Association for College Admission Counseling (NACAC). Special report on the transfer admission process, April 2010.

[50] Navigli, Roberto. Word sense disambiguation: A survey. *ACM Computing Surveys 42* (2009), 1–69.

[51] Ni, Yuan, Zhang, Lei, Qiu, Zhaoming, and Chen, Wang. Enhancing the open-domain classification of named entity using linked open data. In *Proceedings of the 9th Internation Conference on the Semantic Web* (2010), pp. 566–581.

[52] Peter, Katharin, and Cataldi, Emily Forrest. *The Road Less Traveled? Students Who Enroll in Multiple Institutions.* NCES 2005–157. Institute of Education Sciences, U.S. Department of Education, 2005.

[53] Ponzetto, Simone Paolo, and Strube, Michael. Knowledge derived from Wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research 30* (October 2007), 181–212.

[54] Porter, Martin F. Snowball: A language for stemming algorithms, October 2001.

[55] Rada, Roy, Mili, Hafedh, Bicknell, Ellen, and Blettner, Maria. Development and application of a metric to semantic nets. *IEEE Transactions on Systems, Man and Cybernetics 19*, 1 (Jan/Feb 1989), 17–30.

[56] Resnik, Philip. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence* (San Francisco, CA, USA, 1995), vol. 1 of *IJCAI'95*, Morgan Kaufmann Publishers Inc., pp. 448–453.

[57] Sahami, Mehran, and Heilman, Timothy D. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th International Conference on the World Wide Web* (New York, NY, USA, 2006), ACM, pp. 377–386.

[58] Salton, Gerard, and Buckley, Christopher. Term weighting approaches in automatic text retrieval. *Information Processing and Management 24* (August 1988), 513–523.

[59] Seco, Nuno, Veale, Tony, and Hayes, Jer. An intrinsic information content metric for semantic similarity in Wordnet. In *Proceedings of the 16th European Conference on AI* (2004).

[60] Shawe-Taylor, John, and Cristianini, Nello. *Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[61] Turney, Peter D. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *ECML* (2001), Luc De Raedt and Peter A. Flach, Eds., vol. 2167 of *Lecture Notes in Computer Science*, Springer, pp. 491–502.

[62] Wagner, Christian. Breaking the knowledge acquisition bottleneck through conversational knowledge management. *IRMJ 19*, 1 (2006), 70–83.

[63] Waterman, D. A. *A guide to expert systems*, 1st ed. Addison-Wesley, Reading, Mass., 1986.

[64] Witten, I. H, Frank, Eibe, and Hall, Mark A. *Data mining: practical machine learning tools and techniques*, 3rd ed ed. Morgan Kaufmann, Burlington, MA, 2011.

[65] Wu, Fei. *Machine Reading: from Wikipedia to the Web.* PhD thesis, University of Washington, 2010.

[66] Wu, Zhibiao, and Palmer, Martha. Verb semantics and lexical selection. In *Proceedings 32nd Annual Meeting on Association for Computational Linguistics* (1994), pp. 133–138.

[67] Yang, Beibei, and Heines, Jesse M. Using semantic distance to automatically suggest transfer course equivalencies. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications* (2011), pp. 142–151.

[68] Yang, Beibei, and Heines, Jesse M. Domain-specific semantic relatedness from Wikipedia: Can a course be transferred? In *Proceedings of the NAACL HLT 2012 Student Research Workshop* (Montréal, Canada, June 2012), Association for Computational Linguistics, pp. 35–40.

[69] Yang, Dongqiang, and Powers, David M. W. Measuring semantic similarity in the taxonomy of wordnet. In *Proceedings of the 28th Australasian Conference on Computer Science* (Darlinghurst, Australia, 2005), vol. 38, Australian Computer Society, Inc., pp. 315–322.

[70] Yu, Jonathan, Thom, James A., and Tam, Audrey. Ontology evaluation using Wikipedia categories for browsing. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management* (2007), pp. 223–232.

[71] Zesch, Torsten, Müller, Christof, and Gurevych, Iryna. Using Wiktionary for computing semantic relatedness. In *Proceedings of the 23rd National Conference on AI, Vol. 2* (2008), pp. 861–866.

INDEX