DOMAIN-SPECIFIC KNOWLEDGE EXPLORATION WITH ONTOLOGY HIERARCHICAL RE-RANKING AND ADAPTIVE LEARNING AND EXTENSION

by

GRACE ZHAO

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2018

Domain-Specific Knowledge Exploration with Ontology Hierarchical Re-Ranking and Adaptive Learning and Extension

by

Grace Zhao

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Professor Xiaowen Zhang

Date	Chair of the Examining Committee

Professor Robert M. Haralick

Date	Executive Officer

Supervisory Committee:

Professor Xiaowen Zhang

Professor Theodore Brown

Professor Xiangdong Li

Professor Lixin Tao

THE CITY UNIVERSITY OF NEW YORK

Abstract

Domain-Specific Knowledge Exploration with Ontology Hierarchical Re-Ranking and Adaptive Learning and Extension

by

Grace Zhao

Advisor: Professor Xiaowen Zhang

The goal of this research project is the realization of an artificial intelligence-driven lightweight domain knowledge search framework that returns a domain knowledge structure upon request with highly relevant web resources via a set of domain-centric re-ranking algorithms and adaptive ontology learning models. The re-ranking algorithm, a necessary mechanism to counter-play the heterogeneity and unstructured nature of web data, uses augmented queries and a hierarchical taxonomic structure to get further insight into the initial search results obtained from credited generic search engines. A semantic weight scale is applied to each node in the ontology graph and in turn generates a matrix of aggregated link relation scores that is used to compute the likely semantic correspondence between nodes and documents. Bootstrapped with a light-weight seed domain ontology, the theoretical platform focuses on the core back-end building blocks, employing two supervised automated learning models as well as semi-automated verification processes to progressively enhance, prune, and inspect the domain ontology to formulate a growing, up-to-date, and veritable system.

The framework provides an in-depth knowledge search platform and enhances user knowledge acquisition experience. With minimum footprint, the system stores only necessary metadata of possible domain knowledge searches, in order to provide fast fetching and caching. In addition, the re-ranking and ontology learning processes can be operated offline or in a preprocessing stage, the system therefore carries no significant overhead at runtime.

Dedication

To

my dad

Acknowledgements

I pay my hearty gratitude to those who helped me come to this day. My adviser, Professor Xiaowen Zhang, comes first on the list, whose relentless patience, encouragement, and guidance, have nourished my growth in both academic professionalism and maturity. His attentiveness to details, fascination with new knowledge and challenges, and down-to-earth attitude have set an example for research excellence. His forever enthusiastic attitude and generous heart have inspired me along the way. I would also like to especially thank my committee members, Professor Brown, Professor Li, and Professor Tao, for their valuable advice and support. Their criticism reminded me of overlooked details.

Over the years, so many intellectual minds and gentle hearts have elevated my spirit and nudged me back on track when I faltered or diverged from the right direction. I thank Professor Fredman for her insightful advisement and Professor Yanofsky for his witty and fascinating mind (clarity and simplicity rooted in mathematical sophistication), continuous encouragement, and bountiful kindness. My friend Dr. Bennett has been my cheerleader ever since I set off from the starting line. Last but not least, my dear family's support and love are forever the sunshine and breezes in my world.

Contents

List of Tables

List of Figures

List of Abbreviations

AQE **A**ugmented **Q**uery **E**ngine

DCC **D**omain **C**haracteristic **C**lassifier

DIR **D**omain **I**nformation **R**ichness

DPDE **D**ata **P**rocessing and **D**ispatch **E**ngine

DLKS **D**ynamic **L**ightweight **K**nowledge **S**earch

OCMS **O**ntology **C**ontent **M**anagement **S**ystem

OLE **O**ntology **L**earning **E**ngine

WAY **W**e **A**re in **Y**ou

YAU **Y**ou **A**re in **U**s

Chapter 1

INTRODUCTION

This dissertation defines a theoretical framework of domain-specific knowledge search. It details a set of innovative algorithms that re-rank the search results to bring them under the roof of the knowledge domain of quest, as well as provide sustainable domain ontology learning and building.

1.1 Motivation

One of the research areas in web searching is the "intention," or the context of a search query. It all boils down to one perspective: the knowledge domain of the query string. The focal point of a search engine's ranking weights has been placed around the documents' relevancy, authority, and popularity. The efforts result in highly query-relevant, authoritative, and popular search results, although these may not all be relevant to the knowledge domain that the user is interested in.

An academic researcher once said to me that it was very difficult for him to conduct research on his subject – American History, because it seems that all the search platforms, such as Google Scholar, embed a cross-domain nature, that not only he has to dig into domain topics piece by piece but he also needs to filter through the search results for his domain related articles.

Over the years, as attention has shifted from a broad-based search to specific verticals, some domain-specific search engines have emerged. One of the possible drawbacks of the domain-specific search engines is a limited indexed data bank that is comparable to that of a general search engine system.

Another possible limitation of a generic search platform is that the search result sets can only serve as a quick lookup and narration of the query term, phrase, or sentence. In order to know more related information, the user has to conduct a few more rounds of searches using different query strings. The user may or may not know the exact relationship among the information he has gathered.

The proposed framework, Dynamic Lightweight Knowledge Search (DLKS), brings the generic search a step further, to serve the niche needs of users who request a more in-depth search and learning. In short, a generic search transforms scattered raw data into a set of related and ranked information. Whereas the proposed framework presents the users with clusters of information that are tightly coupled and organized upon relationships in the knowledge domain among clusters. In other words, the system intends to bring users not only the research results of related query but the

knowledge structure and in-depth resources in the domain of interest.

1.2 The Problem Explained

A knowledge domain is the content of a particular field of knowledge.

In order to build a knowledge domain, a domain ontology (or a set of ontologies) is critical and essential, for it is the veins and channels of the domain system. Using machine learning or statistical techniques to build an ontology from the ground up is theoretical rather than practical. For example, DBPedia uses a hand-generated mapping to organize the ontology graph entries.

A domain ontology is usually studded with jargons and vernaculars, which are domain-specific and obscure in nature. A general linguistic (synonym, hyponym, hypernym, etc) mapping and parsing may not be effectual in this case. Thus it would be more effective and less error-prone to have the initial domain ontology manually crafted by the domain experts or at least someone with sufficient understanding of the domain knowledge. Once we have the seed ontology, [1] the ontology enhancement task can possibly be delegated to the computers if a logical and verifiable process is in place. That was the motivation for the research presented in this dissertation.

To define a feasible scope of our study, we restricted the domain-specific ontol-

[1] This is an umbrella term to indicate an ontology or a set of ontologies initially created in a domain space.

ogy to a taxonomic hierarchical structure (graph), to reduce ambiguity, complexity and increase measurability. Our system supported three types of node relationship (binary predicate): IS-A (subclass), INSTANCE-OF (object, individual, leaf), and HAS-A (composition relationship).

With the initial ontology in hand, we set off to collect data to feed the system. For that we created our re-ranker based on three information degrees of the ontology structure: *generality*, *granularity*, and *diversity*. In order to keep the system evolving and stay updated, we applied machine learning techniques to refine and augment the domain ontologies.

Unlike the popular ontology learning approaches, such as lexico-syntactic pattern match, Part-Of-Speech (POS) tagging, and phrase chunking, which rely on a linguistic analysis and filtration, we sought the vestiges of a hidden relationship between a new concept and the seed ontology, via a kernel vocabulary term, from which the new concept was discovered. We called the kernel term the *reference term*. Through analyzing the reference term and its neighboring terms on the ontology graph, as well as the web data of the new concept and the ontology dictionary, we deduced a mapping relationship between the new concept and the seed ontology. In this dissertation, term and concept are interchangeable.

The contribution of the dissertation includes but is not limited to our novel domain-specific re-ranking algorithm, and two supervised prediction models for hierarchical ontology extension – WAY ("We Are in You") Neural Networks and YAU

("You Are in Us") Naïve-Bayes. Our experiment data has shown promising results.

In this dissertation, the terms "web document," "document," and "web page" are interchangeable. Though our re-ranker and our prototype system can be applied to both textual and multimedia web documents processing, we focus our research on textual web documents, since textual resources tend to contain more researchable data (textual content + metadata) in comparison to multimedia resources (mainly metadata).

1.3 Research Objectives

The research explores two areas, re-ranker and ontology learning systems, in order to provide clean, clear, and implementable solutions. It lays out an abstract framework of knowledge search, to make it possible to build the actual applications.

1.4 Chapter Layout

This dissertation is organized into seven chapters as follows:

- Chapter 1: Introduction to the Dissertation Topic
- Chapter 2: Core Concepts and Related Work
- Chapter 3: Mathematical Notations and the Preliminaries
- Chapter 4: The Re-Ranking Algorithm

- Chapter 5: The Ontology Learning Models
- Chapter 6: The DLKS Framework
- Chapter 7: Conclusion and Future Directions

Chapter 2

CORE CONCEPTS AND RELATED WORK

2.1 Core Concepts

2.1.1 Domain Ontology and Domain Knowledge

Domain ontology is a collection of vocabularies and the specifications of the conceptualization of a given domain (Gruber, 1993). [1]

Domain knowledge or domain-specific knowledge refers to an area of human endeavor, or knowledge around a specialized discipline. [2] A knowledge domain specifies such an area.

[1] https://www.igi-global.com/dictionary/ontology-based-multimodal-language-learning/8239

[2] https://en.wikipedia.org/wiki/Domain_knowledge

In philosophy, ontology is the study of what exists. In Artificial Intelligence (AI), an ontology is a specification of the meanings of the symbols in an information system. That is, it is a specification of a conceptualization. It is a specification of what individuals and relationships are assumed to exist and what terminology is used for them. Typically, it specifies what types of individuals will be modeled, specifies what properties will be used, and gives some axioms that restrict the use of that vocabulary. [3]

Ontologies are usually written independently of a particular application and often involve a community which agrees on the meanings of symbols. An ontology consists of:

- a vocabulary of the categories of the things (both classes and properties) that a knowledge base may want to represent;
- an organization of the categories, for example into an inheritance hierarchy using subClassOf or subPropertyOf, or using Aristotelian definitions; and
- a set of axioms restricting the meanings of some of the symbols to better reflect their meaning - for example, that some property is transitive, or that the domain and range are restricted, or that there are some restriction on the number of values a property can take for each individual.

Ontologies are considered one of the pillars of the Semantic Web (SW) technologies. SW technologies went through ups and downs since its inception in the late

[3] `http://artint.info/html/ArtInt_316.html`

90's. Ontology however, has been continuously gaining keen interest from both the industry and academia.

An ontology is a formal knowledge description of concepts and their relationships (Lee et al., 2014). Ontologies, sometimes called "concept maps," are composed at least by classes (concepts of the domain), relations (properties, attributes) and instances (individuals). They play an important role in a *knowledge-based system*, a system that is able to find implicit consequences of its explicitly represented knowledge (Baader, 2003). In order to build an ontology, a well-defined lexicon and logics system, such as an ontology language, taxonomy/metadata system, and vocabulary, has to be in place in order to safeguard the validity and soundness of the ontology. A SW *vocabulary* can be considered as a special form of ontology, usually light-weight, or sometimes as a collection of URIs with a described meaning.

Though an ontology is typically written in RDF [4]-based languages such as Resource Description Framework Schema (RDFS), [5] Web Ontology Language (OWL), [6] the academia and the industry did not embrace the formal languages in defining an ontology, due to their XML-based overly strict syntactic complexity. In recent years, simplified and less strict language structure has emerged to leverage the situation.

[4]The Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications. It is a model for encoding semantic relationships between items of data so that these relationships can be interpreted computationally.

[5]RDF Schema (Resource Description Framework Schema, variously abbreviated as RDFS, RDF(S), RDF-S, or RDF/S) is a set of classes with certain properties using the RDF extensible knowledge representation data model, providing basic elements for the description of ontologies.

[6]The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. It is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs.

The most prominent front-runner is OBO Flat File Format.

The Open Biomedical Ontologies (OBO) repository is a large library of ontologies from the biomedical domain hosted by the National Center for Biomedical Ontology (NCBO). The majority of the ontologies in that repository are written in OBO Flat File Format — an ontology language originally designed for the Gene Ontology (GO). This language (from now on called simply OBO) uses a simple textual syntax that was designed to be compact, readable by humans, and easy to parse. The OBO community has dedicated significant effort to developing tools such as OBO-Edit — an integrated OBO editor and reasoner (Golbreich et al., 2007).

Studer et al. (Studer, Benjamins, and Fensel, 1998) defined four types of ontologies:

- Domain ontologies: describe the vocabulary related to a generic domain by specializing the concepts introduced in the top-level ontology.
- Generic ontologies: provide supertheories, like knowledge about IS-A or PART-OF relation. It is valid across several domains.
- Application ontologies: contain all the necessary knowledge for modeling a particular domain (usually a combination of domain and method ontologies).
- Representation ontologies: provide representational entities without stating what should be represented (e.g. Frame Ontology) and do not refer to any particular domain. For example the Frame Ontology defines concepts such as

frames, slots and slot constraints allowing expressing knowledge in an object-oriented or framebased way.

Coakes suggested one more type (Coakes, 2003):

- Task ontologies: describe the vocabulary related to a generic task or activity by specializing the top-level ontologies.

The web has become a valuable source of information for almost every possible domain of knowledge. This has motivated researchers to start considering the Web as a valid repository for Information Retrieval (IR) and *knowledge acquisition* [7] to define the rules and ontologies required for knowledge.

Knowledge is the sum of what is known: the body of truth, information, and principles acquired by humankind (merriam-webster.com).

2.1.2 Ontology Engineering

Ontology engineering is the formal representations of a set of concepts within a domain and the relationships between those concepts. [8] *Ontology learning* is defined as the set of methods used for building from scratch, enriching or adapting an existing ontology in a semi-automatic fashion using heterogeneous information sources. This data-driven procedure uses text, electronic dictionaries, linguistic ontologies (WordNet [9] for example) and structured and semi-structured information to

[7] Knowledge acquisition is the process of extracting, structuring and organizing knowledge (`http://engineering.purdue.edu`).

[8] `https://en.wikipedia.org/wiki/Ontology_engineering`

[9] WordNet is a widely used broad coverage semantic/lexical network about English.)

acquire knowledge.

Ontology learning is the automatic or semi-automatic creation process of ontologies. Concept hierarchy extension is an incremental learning process to extend the taxonomic structure of an existing ontology with further concepts.

2.1.3 Supervised Learning and Classification Algorithms

Machine learning, which sprang up from Artificial Intelligence (AI), uses statistical techniques to give computer systems the ability to progressively improve performance on a specific task with data. It features Supervised Learning and Unsupervised Learning.

A supervised learning algorithm assigns text to the objects based on features. It analyzes the training data and produces an inferred function, which can be used for mapping new data. While unsupervised learning is to group objects based on similar features or distance, the goal of any supervised learning algorithm is to find a function that best maps a set of inputs to their correct output. To express more formally, a supervised learning model approximates the relationship f between the input data X and the desired output Y.

$$Y = f(X) + \epsilon,$$

where ϵ represents *irreducible* error in the model, which is a theoretical limit around the performance of the algorithm due to inherent noise in the phenomena to be stud-

ied and explained.

Classification is considered a type of supervised learning. It characterizes objects or data into groups by one or more features in order to make data-driven predictions or decisions through building a model from sample inputs. The observations are analyzed into a set of *features* or *independent variables* in statistics, which can be categorical, ordinal, or real-valued.

A *classifier* is a set of algorithms or mathematical model that maps input data to a category.

Artificial Neural Networks and *Naïve Bayes classifier* are two of the most studied and often the best solutions to issues resorting to classifications.

Artificial Neural Networks

The artificial neural networks model is highly related to the design of autonomous machine intelligence. It is a biologically-inspired programming paradigm. Its input nodes of different features correspond to the "neurons" in the neural system. The synapses between nerve cells are represented by "synaptic weights" connecting nodes between layers (input layer, hidden layer, and output layer). The loss function depends on the adaptive parameters (biases and synaptic weights) in the neural network. Neural network methods belong to quadratic classifiers.

Learning rule The learning rule is a rule or an algorithm which modifies the parameters of the neural network, in order for a given input to the network to produce a favored output. It is also called "step function." This learning process accumulatively modifies the weights and bias levels within the network.

Least Mean Square (LMS) rule and gradient decent learning are among the mostly applied learning rule sets.

Activation function Also known as Transfer Function, it maps the resulting values in between 0 to 1 or -1 to 1 depending upon the function.

Sigmoid function (Fig. 2.1) is a non-linear S-shaped activation function, with a value between 0 and 1. It is especially used for models that are to predict the probability as an output.

Sigmoid function Σ is defined as:

$$\Sigma(x) = \frac{1}{1 + e^{-x}}.$$

Just as the neurons perform signal forward propagation and backward propagation, artificial neural networks kick off *feedforward propagation* by default, and can add *backpropogation* on top of that.

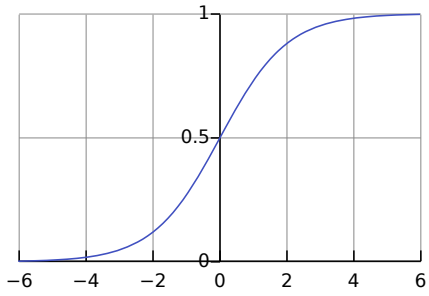

Figure 2.1: Sigmoid Diagram – Logistic Curve

Feedforward propagation In this type of propagation (See Fig. 2.2), the information moves in only one direction, forward, from the input nodes, through the hidden layer(s) (if any) to the output layer. There are no cycles or loops in the network. Therefore, it doesn't result in recurrent neural networks.

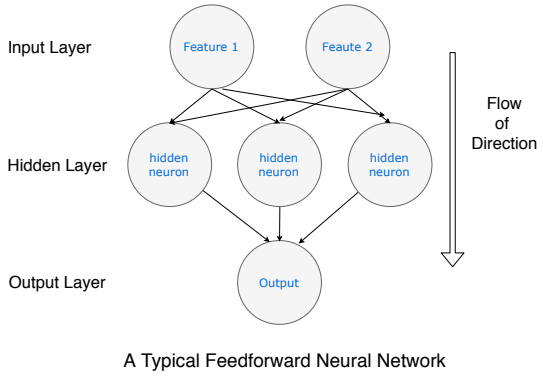

Figure 2.2: Feedforward Propagation in Its Simplest Form

The hidden layers will learn different synaptic weights and hence different functions when fed with the same data. The output layer usually contains the predicted variables that carry class labels. If only a single neuron and no hidden layer, this network would only be able to learn linear decision boundaries. To learn non-linear

decision boundaries when classifying the output, multiple neurons are required. By learning different functions approximating the output dataset, the hidden layers are able to reduce the dimensionality of the data as well as identify mode complex representations of the input data.

Backpropagation The motivation for backpropagation is to train a multi-layered neural network such that it can learn the appropriate internal representations to allow it to learn any arbitrary mapping of input to output. The algorithm can quickly reach the ideal low cost rate. However, gradient descent with backpropagation is not guaranteed to find the global minimum of the error function, but only a local minimum. Besides, the backpropagation is prone to overfitting (wikipedia).

Naïve Bayes

Bayes' theorem is a formula that describes how to update the probabilities of hypotheses (H) when given evidence (E). It follows the axioms of *conditional probability*.

Given a hypothesis and evidence, Bayes' theorem states that the relationship between the probability of the hypothesis before getting the evidence and the probability of the hypothesis after getting the evidence $P(H|E)$ is:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}.$$

Based on applying Bayes' theorem, naïve Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature,

a "naïve" assumption of strong independence between features.

Given a class variable y and a vector of features $\mathbf{x}$, naïve Bayes algorithm states the following relationship:

$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{P(\mathbf{x})}.$$

Let $\hat{y}$ be the label of class y, the naïve Bayes classifer can be written as:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i|y).$$

Naïve Bayes classifiers are highly scalable, requiring a number of parameters being linear in the number of variables (features/predictors) in a learning problem.

Gaussian naive Bayes When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution.

2.1.4 Search Engine Ranking

A search engine is typically composed of a *crawler*, an *indexer* and a *ranker*. The web crawler, also called *spider*, discovers and gathers websites and webpages on the Web. An indexer uses various methods to index the contents of a website or of the Internet as a whole. In order to process the query data and provide a relevant result set, the ranking engine is the "brain" of the search engine. Ranking algorithms work

closely with the indexed data and metadata.

Other than the crawler-based search engine, human-powered directories such as the Yahoo directory, are dependent upon the human editors' manual effort and discretion to build its listing and search database.

There are three categories of web search queries: [10] transactional, informational, and navigational. These are often called "do, know, go." Ranking algorithms often work with informational queries. Query topics can be *broad* or *narrow*. The former pertains to "topics for which there is an abundance of information on the Web, sometimes as many as millions of relevant resources (with varying degrees of relevance) (Lempel and Moran, 2001)." Narrow topic queries refer to those for which very few resources exist on the Web. Therefore, it may require different techniques to handle each characteristic queries.

Search Engine Persuasion (Marchiori, 1997) means "there may be millions of sites pertaining in some manner to broad-topic queries, but most users will only browse through the first k (e.g.: 10) results returned by the search engine." Therefore, the search engine ranking greatly influences the pageview [11] and hit rate [12] of an article on the web.

[10] `http://en.wikipedia.org/wiki/Web_search_query`

[11] A pageview is each time a visitor views a page on your website, regardless of how many hits are generated.

[12] A hit is a request for a file made by a user-agent.

Search engine ranking refers to "the position at which a particular site appears in the results of a search engine query. A site is said to have a high ranking when it appears at or near the top of the list of results." [13]

The explosive growth of web content and widespread accessibility have led to exponential usages of search on the Internet. The users demand higher accuracy and relevancy of search results, which have led to a surge of research activities on search engine re-ranking, aiming to reduce initial search result noise.

Query expansion refers to the process of reformulating a seed query to improve retrieval performance in information retrieval operations. Query expansion involves techniques such as finding synonyms of words, and searching for the synonyms as well; finding all the various morphological forms of words by stemming each word in the search query; fixing spelling errors and automatically searching for the corrected form or suggesting it in the results; and re-weighing the terms in the original query. [14] All general search engines nowadays have already incorporated query expansion in their ranking algorithms. However, the synonyms the general search engines collect may not reflect the synonyms in a domain specific context.

Query augmentation on the other hand refers to adding additional semantic keywords to the search query strings in order to assemble a more relevant search result set.

[13] `http://www.web1marketing.com/glossary.php?term=search+engine+ranking`
[14] `https://en.wikipedia.org/wiki/Query_expansion`

2.1.5 Software Framework

A multi-tier software framework is a blueprint for software implementations. The framework can be theoretical and abstract, or detailed and concrete with implementation environments and technologies. A typical three-tier client-server web application architecture includes *presentation*, *logic*, and *data* tiers.

- Presentation tier This tier launches communications between the application server and the web clients, providing web services and dealing with responses and requests via HTTP(S) protocols.

- Logic tier Also called *application tier* or *middle tier*, the logic tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing.

  This is the most important tier in the framework since it encapsulates the core functions and potency that the application has to offer, and is the hub and control center of communications among all tiers.

- Data tier The tier specifies the communication between application and the database. Other than providing access layer APIs to the application, more database-specific functionality such persistence mechanisms, scalability and maintainability improvement mechanisms can be specified here.

Wikipedia gives a wonderful illustration of the above architecture. [15]

[15] `https://en.wikipedia.org/wiki/Multitier_architecture`

2.2 A Survey of Related Work

2.2.1 Semantic Similarity

Semantic similarity is a metric defined over a set of documents or terms, where the distance between them is based on the likeness of their meaning or semantic content. Computationally, semantic similarity can be estimated by defining a topological similarity, by using ontologies to define the distance between terms/concepts.

In general, there are two types of measures of semantic similarity: 1) based on *information content* (IC), [16] and 2) based on path length. The earlier implementation of the two approaches were mostly based on WordNet corpus.

The IC is usually calculated as the negative logarithm of the probability (observed frequency counts) of that concept:

$$IC(c) = -logP(c).$$

Resnik (Resnik et al., 1999), Lin (Lin, 1998a) and Jiang & Conrath (Jiang and Conrath, 1997) have different implementations of the first approach. While Leacock & Chodorow (Leacock and Chodorow, 1998) and Wu & Palmer (Wu and Palmer, 1994) proposed algorithms of the second approach.

[16] In linguistics and information theory, the term information content refers to the amount of information conveyed by a particular unit of language in a particular context.

Resnic agreed that the path-length approach was a "natural, time-honored way to evaluate semantic similarity in a taxonomy." However, he pointed out that the approach limited itself to uniform distances. Therefore, he proposed his IC-based similarity measure of two concepts, c_1 and c_2, as:

$$sim(c_1, c_2) = max_{c \in S(c_1, c_2)}[-logP(c)],$$

where $S(c_1, c_2)$ is the set of concepts that subsume both c_1 and c_2. A class that achieves the maximum value in the equation is termed as ***most informative subsumer***.

Leacock & Chodorow used path length to exploit semantic similarity between two words in the WordNet:

$$Sim_{ab} = max[-log(Np/2D)],$$

where Np is the number of nodes in the path p from a to b, and D is the maximum depth of the taxonomy.

2.2.2 Web Document Re-ranking

Many researchers have explored web document re-ranking within a domain structure using different approaches, such as link structure analysis (Finkelstein et al., 2001; Tam and Shepherd, 2010), document comparison (Krikon, Kurland, and Bendersky, 2010; Liu et al., 2004), learning (Burges, 2007; Huang and Hu, 2009; Kang

et al., 2011), and recommendation systems (Li, 2011), among others. Here, we will only briefly introduce some prior research related to domain-specific re-ranking algorithms. In particular, we will mention some works that discuss the notions of "augmented search query," "generality," "granularity," and "diversity" which are some of the core concepts in our algorithm formation.

Augmented Query

Using augmented search queries is not new. In the intelliZap system (Finkelstein et al., 2001), its algorithm utilizes the semantic network to extract keywords from the context surrounding the user-selected text. These keywords are added to the text to form an augmented query, leading to context-guided information retrieval. The system suggests a domain classification algorithm to identify the domain of the query context. It is based on probabilistic analysis and classifies the context to a limited number of high-level domains (e.g., medicine or law) by determining the amount of similarity between predefined domain "signatures" and the query context. In order to compute the domain signature, a corpus of approximately 100,000 words is sampled for each domain.

The probability of a domain given a particular text query, $P(Domain_j|Text)$, can be represented according to Bayes' rule as follows:

$$P(Domain_j|Text) = \frac{P(Text|Domain_j)P(Domain_j)}{P(Text)}.$$

However, the process of extracting keywords, performing a domain match and

searching through an array of domain-specific and general purpose search engines, then performing the re-ranking order, seems to be lengthy and unfeasible.

Generality

The word "generality" holds different connotations when it comes to web re-ranking algorithms. Yan et al. (Yan, Li, and Song, 2006) defined overall document generality as "broad enough to cover as many different aspects of a certain topic as possible." Aiming to improve the query performance of domain specific (bio-medical literature in this paper) information retrieval by re-ranking retrieved documents on generality, their re-ranking algorithm was highlighted by computing a combined score of similarity and generality of the document related to the query. Other than proposing two types of generality ranking: query generality and document generality, the paper suggests a conceptual marking tree – the domain related ontology tree – to define a generality scope. The final generality score is calculated based on the concept of document cohesion – the state or quality that the elements of a text "hang together," and document scope – how broad or vague a document is for describing a certain topic. The algorithms are defined as follows:

$$Scope(d_i) = e^{-\frac{\Sigma_{i=1}^{n} depth(c_i)}{n}},$$

where n is the total number of concepts of both the ontology concepts and general concepts. $depth(c_i)$ is the distance between concept i and the root of the ontology tree.

$$Cohesion(d_i) = \frac{\Sigma_{i,j=1}^{n} Sim(c_i, c_j)}{Numberof Associations}, (n > 1, i < j).$$

$$Sim(c_i, c_j) = -log\frac{len(c_i, c_j)}{2D},$$

where D is the ontology tree maximum depth. $len(c_i, c_j)$ is the shortest path between concept i and j in the ontology tree.

$$Numberof Associations = \frac{n(n-1)}{2},$$

where n is the total number of ontology concepts in a document d_i.

$$generalityScore(d_i) = \frac{Scope(d_i)}{Cohesion(d_i) + 1}.$$

Allen and Wu (Allen and Wu, 2002) measured document generality based on the mean generality of domain concepts contained in a document. The notion "generality" was defined as "general things or concepts," which are easier for individuals to understand than are jargon-filled technical documents. They state that their measurement based on WordNet ontology can detect the more general term in a hierarchy structure. For instance, the term "animal" was computed as a more "general" term compared to the term "dog."

Granularity

A concept-based computational algorithm was developed to estimate the "semantic granularity" of documents with reference to domain ontology, along with "similarity" and "popularity" measures. Yan et al. (Yan et al., 2011) defined semantic granularity as "the levels of semantic detail carried by an information item."

The proposal of *subtopical retrieval* (Zhai, Cohen, and Lafferty, 2003) was an automatic granularity-based document ranking algorithm. It argued that, in cases such as a literature survey, documents need to be found that cover as many different subtopics of a general topic as possible.

Diversity

In Affinity Ranking (Liu et al., 2004), the authors addressed information diversity according to two aspects: the topic coverage of a group of documents and the amount of information contained in a document.

Huang and Hu (Huang and Hu, 2009) promoted information diversity in their re-ranking algorithm in the biomedicine domain. The algorithm computes the maximum probability of its hidden properties corresponding to each retrieved passage iteratively until all subsets achieve stability, and then these passages are re-ranked from different subsets.

2.2.3 Ontology Learning

Ontology learning is the creation process of ontologies, including extracting the corresponding domain's terms and encoding them with an ontology language for easy retrieval. There are three main categories of ontology learning: pattern-based, clustering-based, and ontology-based.

The pioneer of pattern-based ontology learning was Hearst (Hearst, 1992), who used a hand crafted list of hyponym patterns as seeds and employed bootstrapping to discover relations. Later on, more pattern-based ontology learning algorithms were proposed (Berland and Charniak, 1999) (Kozareva, Riloff, and Hovy, 2008).

Clustering-based approaches cluster terms hierarchically based on the similarities of their meanings, usually represented by a vector of quantifiable features, which has led to taxonomy relationship discovery. The common types of clustering-based features include similarities of the vectors (Brown et al., 1992), contextual relations (Lin, 1998b), verb-noun relations (Pereira, Tishby, and Lee, 1993), syntactic dependency (Pantel and Ravichandran, 2004), and co-occurrence (Yang and Callan, 2008; Liu et al., 2005).

Ontology-based learning or ontology referencing refers to the knowledge that engineers use open ontologies such as WordNet (Miller, 1995) and CYC Knowledge Base (Lenat, 1995) as their authoritative reference backdrop to create or extend custom ontologies. Liu et al (Liu et al., 2005) describe a learning method that uses co-occurrence analysis and trigger-phrase (pattern) to find hierarchical relationships based on the WordNet lexical dictionary. Dresden Ontology Generator for Directed Acyclic Graphs (DOG4DAG) (Wächter and Schroeder, 2010) is an ontology-based ontology learning system, which supports the creation and extension of ontologies by semi-automatically generating terms, definitions and parent-child relations from text in PubMed [17] and the web. It uses the Ontology Lookup Service (Côté et al., 2006) to

[17]PubMed is a free search engine accessing primarily the MEDLINE database of references and

search for related ontologies and terms, and maps a new term to the ontology at hand.

Many learning systems adopt a heuristic approach of combining two or three above categories (Novalija, Mladenić, and Bradeško, 2011) in ontology engineering. However, the driving force of ontology acquisition and analysis of all categories is mainly centered around the linguistic and conceptual quality of various forms of evidence underlying the generation and refinement of conceptual hypotheses. Our proposed ontology-based algorithms were not leaning towards a linguistic approach. Instead of referencing external ontologies, our algorithms were seeking inward to the seed ontology for all possible traces of taxonomic relations.

Prior researches on domain ontology augmentation using World Wide Web (WWW) resources (Vossen, 2001; Agirre et al., 2000) tended to take a lexicalization approach or resort to lexical ontology WordNet, which was the limitation (lexical dependency) that our research tried to overcome.

For incremental ontology learning processes, Kozareva and Hovy (Kozareva and Hovy, 2010) started with an initial given set of core and basic level concepts, and used Hearst-like lexico-syntactic patterns iteratively to harvest new terms from the Web. As a result, a set of hypernym-hyponym relations was obtained. Pattern-based approaches provide relatively high precision but typically suffer from low recall due to sparse coverage of patterns in a given corpus (Cimiano et al., 2005).

abstracts on life sciences and biomedical topics.

Concept hierarchy extension A type of ontology augmentation, concept hierarchy extension is an incremental learning process to extend the taxonomic structure of an existing ontology with further concepts. The prior researches in the field using ontology-based learning and WWW resources (Vossen, 2001; Agirre et al., 2000) generally utilize a lexicalization approach or resort to lexical ontology WordNet.

The ontology learning process is often combined with machine learning techniques for ontology extraction, knowledge acquisition and term classification, in both supervised (inductive) (Navigli, Velardi, and Gangemi, 2003) and unsupervised (Geetha, 2017) learning. Decision Tree (Zhang et al., 2014) , Bayes Model (Feng et al., 2015; Tang et al., 2016), SVM (D'Orazio et al., 2014), neural networks (Tang, Qin, and Liu, 2015), and NLP techniques (Sabrina, Rosni, and Enyakong, 2001; Hahn and Markó, 2002), are some of the most featured models.

Chapter 3

MATHEMATICAL NOTATIONS AND THE PRELIMINARIES

3.1 Mathematical Notations

Definition of *domain ontology*: Let $G = (V, E, L)$ be a rooted, directed tree (the ontology graph), where V is the set of nodes/vertices in the tree. A node/vertex in G is a term or a set of terms that are synonyms or aliases, which share the same term definition or description. The root node rN is the most abstract term, which is usually titled with the ontology domain name. E is a collection of all edges (arcs) in G. All edges in the graph are labeled. L is a set of labels in G. L = (IS-A, INSTANCE-OF, HAS-A), where IS-A represents subclass relation, INSTANCE-OF indicates an instance relation with the node it's derived from, and HAS-A suggests a composition relationship between two ontologies. If a query string is mapped to some term/node t_{ref} in G, we call t_{ref} the *reference node* or the *reference term*. (For

notation convenience, we also refer t_{ref} as the reference node i.)

We denote the number of all nodes in G as $|V|$ and number of all leaf nodes in G as $|V_{\text{leaf}}|$. The depth (maximum number of back edges) of G is $Depth(G)$. $d(v, rN)$ denotes the shortest distance between some node $v \in V$ and rN.

A new term/node $t_{\text{n}} \notin V$ is to be added to G, along with a new edge e_n. $G' = ((V, t_{\text{n}}), (E, e_{\text{n}}), L)$ is the enhanced ontology graph. For convenience, we use $t'_{\text{n}} : (t_{\text{n}}, e_{\text{n}})$ to denote the proposed nodal position in G for t_{n}.

t_{n} can imply one or more of the following meanings:

- a new vocabulary term,
- a set of its semantic synonyms,
- a rooted tree with the term as the root node.

Definition of *augmented query*: Let Γ be a set of domains, and some generic query q. A domain $\gamma \in \Gamma$, its *domain characteristic classifier* is τ_γ, therefore $\tau_\gamma \in \gamma$. The *augmented query* is $q_\gamma = q \cdot \tau_\gamma$, where $\cdot$ denotes string concatenation, and hence, $q_\gamma \in \gamma$.

Definition of *initial rank*: The initial rank order $R_{init(i)}$ per an augmented query q_γ is a document collection $R_{init(i)} = \{d_m | m = \{1, 2, ..., N\}\}$, where node i in

the ontology graph of domain γ is corresponding to q. N is the number of retrieved documents. Document d_1 ranks the highest, and d_N the lowest.

Link Relations

Link relations among all nodes in G can be represented by a *link relation matrix*, an adjacency $|V| \times |V|$ matrix M. Each entry in M represents the weight of a link between two nodes, i and j. Let $\hat{M}$ be the normalized M, where the sum of each row is 1. Please note: $(i, j) = 0$, if $i = j$. Thus in the matrix $\hat{M}$, all diagonal entries (reflexive) are zeros.

We demarcate the link relations into three major categories (three dimensional information) between two distinct nodes i and j in G: *distance*, *direction*, and *relationship*.

The distance between nodes i and j is $d(i, j)$, a non-negative integer, which is the number of edges between the two nodes along the shortest path. In turn $d(i, j)$ defines the ***distance relation*** between the nodes i and j.

The ***direction relation*** is comprised of the three degrees of information: *generality, granularity, and diversity.* The three degrees are represented in an ontology graph as *parent/ancestor* nodes, *child/descendant* nodes, and *sibling/remote relative* nodes, respectively:

1. *parent/ancestor*: Node j is said to be an *ancestor* of node i, if j is on the (shortest) path from root to i in G, and $j \neq i$. Node j is the *parent* if it is adjacent to node i. The relationship denotes the *generality* degree of node j in reference to node i.

2. *child/descendant*: Node j is said to be a *descendant* of node i, if i is on the (shortest) path from root to j in G, and $j \neq i$. Node j is a *child* of node i if it is adjacent to i. The relationship denotes the *granularity* degree of the node j in reference to node i.

3. *sibling/remote relative*: Nodes i and j are said to be *siblings* if they have a common *parent* in G, and $j \neq i$. The relationship implies *diversity* degree of node j in reference to node i.

   Node j in G is said to be a *remote relative* of node i, if node j is not a *parent/ancestor*, or a *child/descendant*, or a *sibling* of node i. The relationship too implies the *diversity* degree of node j in reference to node i.

See Figure 3.1 3.2 of our gerontology ontology for an example of the aforementioned three *directional* relations.

The ***relationship*** or the ***attributive relation*** concerns the link label properties between nodes m and n. If there is an edge $(m, n) \in E$, the label $l_{(m,n)} \in L$ defines the *attributive relation* between m and n. There are two major labels in our prototyping system: IS-A (a subclass) and INSTANCE-OF (an instance).

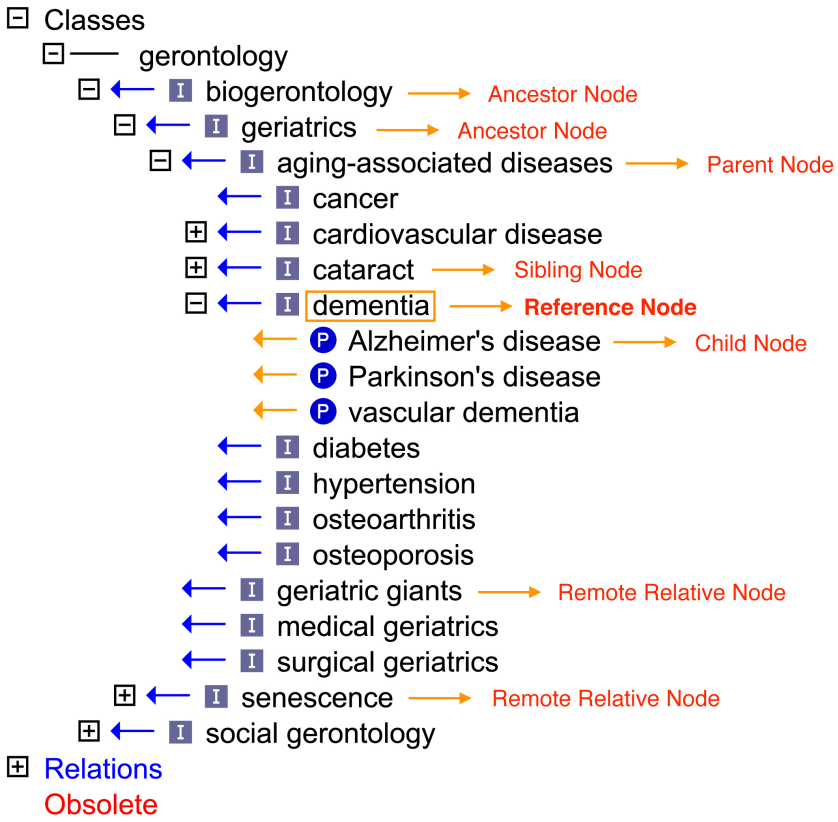

Figure 3.1: Partial View of Gerontology Ontology

An instance of a concept is said to be more specific than a subclass of a concept. Therefore, the *attributive* relation underlines a semantic perspective of *granularity*, while the *directional* granularity implies a structural perspective.

Definition of *parent, sibling, child node*: Let v be a vertex (node) in V, $v \in V$. Let $f_{\mathrm{p}}(v)$ be a function that gets the set of parent nodes of node v, $v \in G \backslash rN$. That is

$$f_{\mathrm{p}}(v) = \{x | \text{ node adjacent to } v \text{ on the path to } rN\}. \tag{3.1}$$

Since G is a tree, $f_{\mathrm{p}}(v)$ returns one node only, which is the **parent node** of v.

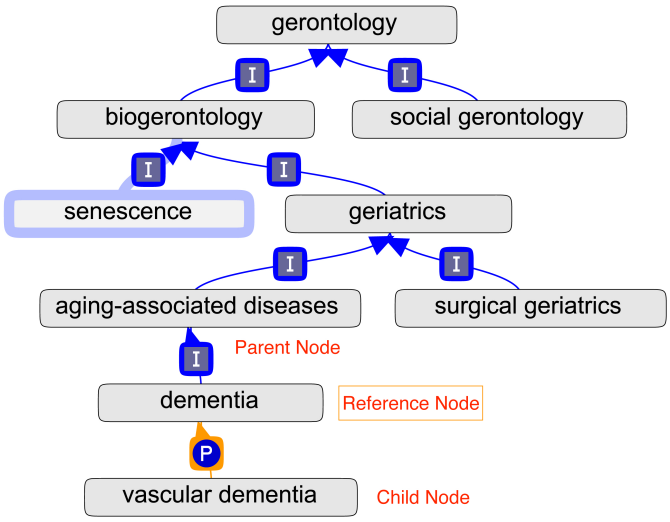

Figure 3.2: Partial View of Gerontology Ontology (Graph View)

The **sibling node set** of v, f_{s}, is defined as:

$$f_{\mathrm{s}}(v) = \{x | f_{\mathrm{p}}(x) = f_{\mathrm{p}}(v)\}. \tag{3.2}$$

Subsequently, the **child node set** of v, f_{c}, is:

$$f_{\mathrm{c}}(v) = \{x | v \in f_{\mathrm{p}}(x)\}. \tag{3.3}$$

3.2 The Preliminaries

3.2.1 Ontology Data Model

A data model is a logic organization of real world objects. We want to set our framework's core components – the ontologies – to follow an efficient and sufficient data model. An object-oriented (OO) data model is characterized by a class hierarchy, which is in line with the taxonomic hierarchical structure of an ontology. Therefore, we adopt the OO data model's core terminologies and structure as our ontologies' base model.

OO model is an abstract data model that does not specify the underlining implementation language, which gives freedom and flexibility for model implementation.

In our research, an ontology is a domain ontology, a tree graph with hierarchical relationships among nodes. The root node is a class. All internal nodes are subclasses. The leaves can be either an instance node or a subclass node. The relationships allowed in the ontology tree are IS-A, INSTANCE-OF (inheritance), and HAS-A (composition).

Data model rules:

- A node is a concept, which can be a class, a subclass, or an instance.
- An IS-A relationship denotes the relationship between a class and its subclass.

- A HAS-A relationship denotes a relationship between a node in one ontology and a node in another ontology.
- Only nodes with an IS-A relationship are native to the domain ontology tree.
- An ontology is a cluster of concepts that all internal nodes and leaf nodes are derived from the root concept.
- Each concept is defined by a name, a description, and a set of URLs (ranked articles).

In this way, we can build the ontology tree with a higher degree of correctness, and it is easy for maintenance. This data model of ontology encourages multiple lightweight ontologies that can be connected with HAS-A relationship under a knowledge domain, instead of compiling a bulky complex network with convoluted concepts and a large number of properties, which results in too complicated relationships that are difficult to identify or verify.

Since there are many overlapping concepts among description logic (DL), first-order logic (FOL) and Web Ontology Language (OWL), the native language to Semantic Web ontology, we list the key terms in each domain that are of the same or similar notation in the following table 3.1.

Table 3.1: Terminology compared among FOL, OWL, DL, and OO

FOL	**OWL**	**DL**	**OO**
constant	individual	individual	instance, object
unary predicate	class	concept	class
binary predicate	property	role	relationship

3.2.2 Semantic Hypothesis

There are primarily three dimensions of information we study between two nodes in an ontology graph: *distance* the least number of hops from one node to another, *direction* the directional relation between two nodes , and *relationship*, represented by edge labels. The *direction* relation is subsequently divided into three degrees: *generality*, *granularity*, and *diversity*. [1]

In a domain-specific locality, information *granularity*, sometimes referred to as *sub-concepts* of the query term, is considered to contain information specialty and depth. Information *generality* provides a more overall view than the query string, in reference to the topic and the domain. *Granularity* and *generality* can be considered to be in close quarters in terms of information *relevance* per the query string. In our algorithm, we distribute more weights to *granularity* degree over *generality*, due to a domain-specific context, with the consideration that an expert in the domain tends to possess more in-depth knowledge. In Information Retrieval theory, it's believed that the more abstract a concept (in a taxonomy-like structure), the lower its IC. Information *diversity* refers to the amount of information that is related to the query term, but it is neither more *general* nor more *granular* in the domain structure. We

[1] We initially studied the related semantic hypothesis in (Zhao, Zhang, and Tansel, 2014).

assign the *diversity* weight mid-way between *generality* and granularity – The supposition is that information *diversity* may speak with a more cohesive quality of the domain space for the document than does its *generality* counterparts, but may also contain less comprehensive information than do the *granularity* properties.

We compute a comprehensive score that incorporates the weights of three dimensions of information and the three degrees of information. We call the score the *Domain Information Richness (DIR).*

3.2.3 Priori – Augmented Search Queries

Let us select one word, or a short phrase that characteristically represents the domain being discussed, and we then add it to the query string. In doing so, we semantically and "dimensionally" narrow down the search scope and bring the search results within, or close to the intended knowledge domain. We call this chosen auxiliary term the *domain characteristic classifier (DCC).*

This keyword should not be too jargon-ish, such as "polymorphism" the field of computer science, or "senescence" in the domain of gerontology. It should be as characteristic as it can be, as well as being as familiar and commonplace as possible.

Usually, the domain name can be used as the DCC. If the domain name is not so familiar, such as "gerontology," one can spot such classifier keywords from the definition of the domain name. For example, in our prototyping system, we built a

platform in the domain of gerontology. "Gerontology" is defined as: "The scientific study of old age, the process of aging, and the particular problems of old people. [2]" We could use "old age" or "old people" as the classifier. Since "old people" indicates human race, it may be more characteristic in the gerontology domain than is the term "old age." Nonetheless, any DCC could greatly "domainify" the search results.

We experimented using more than one DCC, and found that too many auxiliary keywords may overpower the original query term, causing the search results to be less relevant to the original query term, but leaning more towards the auxiliary, keywords, relatively. Therefore, we only recommend to use one DCC to form the augmented query. We also observed the subtle difference between returned document sets resulting from different DCCs.

This is the simplest and for that reason the most effective way to roughly bring a search result within a knowledge domain, if we pick an appropriate DCC.

Since a DCC plays an instrumental role in an augmented query search, a rigorous procedure for the selection of such a term becomes necessary. We conclude a few core principles for DCC selection:

- The DCC should be non-ambiguously representing the topical domain, and should be representing the domain knowledge as a whole, but not partially. One of the qualified DCC candidates should be the domain name itself. Another

[2]The Oxford Dictionary of Difficult Words

possible route of picking such a DCC is to scan the keywords from the definition of the domain knowledge.

- The DCC should be terse, with the least amount of words, ideally one word only. This concern is to address the adherence and degree of loyalty to the intended meaning of the original query string. In other words, to keep DCC to its minimum presence, it will not heavily influence the search result, deviating the semantics from the initial search query.

  What if we can not find one DCC that represents a domain to its totality. Can we use more than one DCCs in an augmented query? We experimented using two or more DCCs, and found that using more auxiliary keywords increases the degree of overpowering the original query terms, hence causing the search results to be less likely relevant to the original query search results. Therefore, we recommend to use only one DCC as the augmented query.

- Since we reply on one or a few general-purpose search engines to perform our initial search, a common and familiar word may induce a better search result. Ergo a jargon or a rare word is not preferred or should be avoided from being chosen as a DCC. For example, in our prototyping system, we built a platform in the domain of gerontology. "Gerontology" can be categorized as a "jargon," which is not suitable for a DCC. After studying the definition of gerontology: "The scientific study of old age, the process of aging, and the particular problems of old people [3]," we consider "old age" or "old people" to be representative in

[3]The Oxford Dictionary of Difficult Words

the gerontology domain as a whole and is a daily use phrase, therefore qualified to be a DCC candidate.

The technique looks similar to query expansion, however, with quite different purpose and implementation procedure. Query expansion concentrates on lexical meaning of the seed query string and reformulates the query string to include synonyms, stem-word-based morphological expressions, or simply rephrases the query. We on the other hand focus on bringing the search scope down to a domain space. To avoid confusion, we use query augmentation but not query expansion to illustrate our DCC concept. In fact, most of the general search engines have already implemented query expansion in their search algorithm.

3.2.4 Ontology Hierarchy Extension Rules

The hypothesis of our proposed algorithms is that t'_{n} should be as close as possible to where t_{n} was mined from — t_{ref}.

For that reason, we have the following rules for placing a new node on the ontology graph:

1. A new node t_{n} can be placed below the reference node t_{ref}, as a *child* node of t_{ref}.
2. A new node t_{n} can be placed below the parent node of t_{ref}, as a *sibling* node of t_{ref}.
3. The default edge label of t_{n} is IS-A

The issue at hand becomes a binary classification problem over two classes: SIBLING, denoting a sibling node of t_{ref}; and CHILD, denoting a child node of t_{ref}.

To regulate our ontology learning and to ensure a structured system, we imposed the following rules for ontology construction.

- Only a class/subclass node can extend a child node. An instance node is a leaf node that cannot further obtain a child node.
- A subclass node can have either a child node of IS-A relation (a subclass node) or INSTANCE-OF relation (an instance node, a leaf node).
- A new node can be inserted only once, and only in one place, in the ontology graph.

For node deletion, we restrict it to three operations:

- Delete a leaf
- Delete a cluster (from the cluster root till all its leaves)
- Replace/rename a node

Chapter 4

The RE-RANKING ALGORITHM

4.1 The Algorithm

The re-ranking algorithm is simply carried out to a candidate web page d_m in three major steps:

1. Computation of the *base score* θ based on d_m's original ranking position.
2. Computation of the *Domain Information Richness* score ω of d_m. More specifically, how much information of *distant, directional*, and *attributive* relations between reference node i and any other node j in G does d_m hold?
3. Computation of the new rank R' of d_m.

4.1.1 Base Score

In order to make the Base Score more computation-friendly and with more judicious semantic sense, we bend the uniformly distributed linear line of the initial rank order $R_{init(i)}$ into a decreasing curve with non-uniformed distribution – giving the top k ranked items special concentration, in concave shape, and the rest in convex shape (See Figure 4.1 for illustration). Such operation aligns with *search engine persuasion* [1] moral.

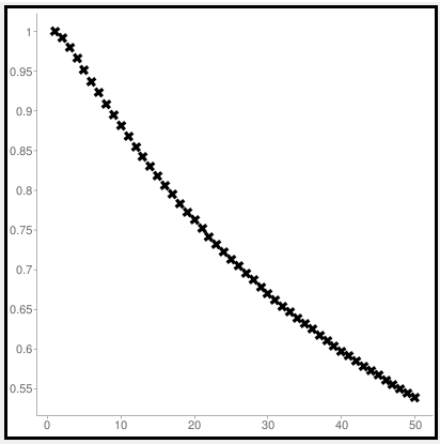

Figure 4.1: Scatter Plot of Base Scores (N=50, b=10)

The *base score* $\theta(m)$ is computed as such:

$$\theta(m) = \begin{cases} 1, & \text{if } m = 1, \\ \frac{N+2log_b(m+1)}{m+N}, & \text{otherwise,} \end{cases} \tag{4.1}$$

where N is the total number of retrieved documents, $1 \leq m \leq N$ is the original rank

[1] Search engine persuasion refers to the phenomenon that there may be millions of sites pertaining in some manner to broad-topic queries, but most users will only browse through the first k (e.g.: 10) results returned by the search engine.

position, and b denotes the base of the logarithm. Presumably, b can be considered the top b articles of high quality in authority and relevance in ranking.

The above formula, partially inspired by Cumulated Gain-Based Measurements (Järvelin and Kekäläinen, 2002), is one of the theoretical contributions of our research.

4.1.2 Domain Information Richness (DIR)

We now compute the *distance*, *direction*, and *relationship* scores of document d_m.

Distance Score

The distance score can be considered as the similarity measure between reference node i and some node j in G, where $i \neq j$. We adopted the popular Leacock Chodorow Similarity algorithm (Leacock and Chodorow, 1998) as the basis of our distance score formula. The basic intuition of the Leacock-Chodorow function is that the semantic similarity between two concepts is estimated based on the conceptual links (i.e., the distance) between these concepts in ontology.

The distance (similarity) score between reference node i and some node j is defined as:

$$sim(i,j) = -log_2 \frac{d(i,j)}{2Depth(G)}, \tag{4.2}$$

where $d(i,j)$ is the distance between nodes i and j.

This is a monotonically decreasing function with respect to the increasing neighborhood distance between *reference node* i and any other node j allotted in G.

Direction Score

We assign weights to the three degrees of information in the dimension of direction relation - *granularity*, *diversity*, and *generality*, based on our hypothesis stated in Section 3.2.2.

Let $\nu_{gr}, \nu_{di}, \nu_{ge}$ be the weights assigned to *granularity*, *diversity*, and *generality*, respectively, where $\nu_{gr} \geq \nu_{di} \geq \nu_{ge}$. For simplicity, let $\nu_i(j)$ denote one of the aforestated directional weights of node j in reference to node i.

In our experiment, we set and tuned the three weights as follows:

$$\nu_i(j) = \begin{cases} \nu_{gr} = 0.15, & \text{if } j \text{ is a } \textit{child/descendant} \text{ of node } i, \\ \nu_{di} = 0.13, & \text{if } j \text{ is a } \textit{sibling/remote relative} \text{ of node } i, \\ \nu_{ge} = 0.11, & \text{if } j \text{ is a } \textit{parent/ancestor} \text{ of node } i. \end{cases}$$

We set the upper bound of the combined weight rates to 40% (0.4), in order to not overpower the original rank order. The total of the above coefficients is $0.15 + 0.13 + 0.11 = 0.39$.

The direction score $drn_i(j)$ between reference node i and some node j is computed

as:

$$drn_i(j) = \nu_i(j) \times sim(i, j). \tag{4.3}$$

Relationship Score

The relationship weight, ν_{re}, should be assigned a fraction of $\nu_i(j)$, since the value is additive to any of the directional weights. We set $\nu_{re} = 0.02$.

The relationship score, $attr(j)$, is defined as:

$$attr(j) = \begin{cases} \nu_{re} \times sim(i, j), & \text{if } j \text{ is an instance node,} \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Link Relation Matrix

With the distance, direction and relationship scores in place, we can compute the square matrix M with the link relation weights.

$$M(i, j) = \begin{cases} 0, & \text{if } i = j, \\ drn_i(j) + attr(j), & \text{otherwise.} \end{cases} \tag{4.5}$$

Let $||M_i||_1 = 1$, where $||M_i||$ denotes the L1 norm of M_i, the ith row in M, then we get a normalized matrix $\hat{M}$.

Note that this matrix (see Algorithm 1) can be pre-calculated and cached before any re-ranking computation takes place.

Algorithm 1: How to Compute Link Relation Matrix

Data: Nodes and (labeled) Edges in G
Result: linkRelationMatrix
[Declaration];
$HashMap\ linkRelationMatrix$;
[Initialization];
$Node\ nodes[V] \leftarrow$ [input from database];
$Depth(G) =$[height of ontology tree];
$\nu_{gr}, \nu_{di}, \nu_{ge}, \nu_{at} \leftarrow$ [weight constants];
while *(loop V: index i)* **do**
    Array row[V];
    while *(loop V: index j)* **do**
        compute $\nu_i(j)$;
        compute $d(i,j)$;
        compute $attr(j)$;
        $row[j] = \nu_i(j) \times (-log_2 \frac{d(i,j)}{2 \times Depth(G)}) + attr(j)$;
    $normalize(row)$;
    $linkRelationMatrix[i] = row$;

Domain Information Richness (DIR) Score

The DIR score measures how much information a single document contains in its topic domain locality. The DIR score of d_m per reference node i is the sum of all *directional* and *attributive* scores mapped to d_m.

Let $\boldsymbol{v}_{i,d_m}$ be a vector composed with 0s and 1s, $|\boldsymbol{v}_{i,d_m}| = |V|$. If d_m contains node j, the corresponding jth element in $\boldsymbol{v}_{i,d_m}$ will be set to 1, 0 otherwise. Let g be the number of 1s in $\boldsymbol{v}_{i,d_m}$.

The initial DIR score of d_m per reference node i is defined as:

$$\omega_i^0(m) = \langle \hat{M}_i, \boldsymbol{v}_{i,d_m} \rangle, \tag{4.6}$$

where $\hat{M}_i$ is the ith row of $\hat{M}$, $\langle \hat{M}_i, \boldsymbol{v_{i,d_m}} \rangle$ is the inner product of vectors $\hat{M}_i$ and $\boldsymbol{v}_{i,d_m}$.

We further define the DIR score of d_m per reference node i as:

$$\omega_i(m) = \begin{cases} 0, & \text{if } g = 0, \\ \omega_i^0(m)/log_b(g+1) \times 10, & \text{if } g \geq 1, \end{cases} \tag{4.7}$$

where b is the base of the logarithm. The $g + 1$ is to avoid the divisor being 0. Due to normalization, the DIR value tends to be far smaller than the *base* score θ, so we increase it tenfold.

We use an inverted index as the coefficient here to penalize a high number of occurrences of matched ontology terms in a document. This could happen to a large document (e.g., a 60-page research article) or a tutorial that may match 20+ ontology terms in its textual body.

4.1.3 Re-Ranking Score

The new ranking score of d_m in domain γ with reference node i (in correlation with the query string) is a linear combination of *base score* θ and the DIR score ω.

$$R'_i(m) = \alpha \times \theta(m) + (1 - \alpha) \times \omega_i(m), \alpha \in [0, 1], \tag{4.8}$$

where α is the adjusting parameter (currently set to 0.85).

4.2 Experiments

4.2.1 Dataset and Queries

We built an experiment system in the domain of gerontology and created the initial prototyping ontology, using OBOEdit software. The ontology language therefore was life science preferred (gene-annotation-initiated) late ontology language: OBO, which stands for Open Biomedical Ontologies. There are 121 terms (concepts) in the graph and the graph depth is seven.

Restricted by our limited resources, we chose three query strings – "fall prevention," "calorie restriction," and "dexterity" – three nodes from the ontology graph, and 30 human evaluators for the experiments. We used "old age" as the DCC at the time of the experiments.

We chose Google as the generic search engine and collected top ten documents of

Table 4.1: Web Documents Used In Our Experiments I

f-p	http://www.who.int/ageing/publications/Falls_prevention7March.pdf
	http://www.who.int/ageing/projects/falls_prevention_older_age/en/
	http://www.cdc.gov/homeandrecreationalsafety/falls/
	http://www.mayoclinic.org/healthy-lifestyle/healthy-aging/in-depth/fall-prevention/art-20047358
	http://nihseniorhealth.gov/falls/aboutfalls/01.html
	https://www.ncoa.org/healthy-aging/falls-prevention/preventing-falls-tips-for-older-adults-and-caregivers/6-steps-to-protect-your-older-loved-one-from-a-fall/
	https://www.ncoa.org/healthy-aging/falls-prevention/preventing-falls-tips-for-older-adults-and-caregivers/debunking-the-myths-of-older-adult-falls/
	http://orthoinfo.aaos.org/topic.cfm%3Ftopic%3Da00135
	http://ageing.oxfordjournals.org/content/35/suppl_2/ii37.short%3Frss%3D1%26ssource%3Dmfc (*no longer available*)
	https://www.nia.nih.gov/health/publication/falls-and-fractures
c-r	http://ajcn.nutrition.org/content/78/3/361.full
	https://www.fightaging.org/archives/2002/11/calorie-restriction-explained/
	http://www.nature.com/articles/ncomms4557
	https://www.nia.nih.gov/newsroom/topics/calorie-restriction
	http://www.ncbi.nlm.nih.gov/pubmed/24691430
	http://www.nih.gov/news-events/news-releases/nih-study-finds-calorie-restriction-lowers-some-risk-factors-age-related-diseases
	https://en.wikipedia.org/wiki/Calorie_restriction
	http://www.lifeextension.com/protocols/lifestyle-longevity/caloric-restriction/Page-01
	http://www.livescience.com/2666-live-longer-anti-aging-trick-works.html
	http://www.crsociety.org/

f-p: fall prevention; c-r: calorie reduction.

Table 4.2: Web Documents Used In Our Experiments II

dx	http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4331509
	http://biomedgerontology.oxfordjournals.org/content/58/2/M146.full.pdf
	http://apk.hhp.ufl.edu/wp-content%5Cuploads/Kornatz.pdf
	http://etheses.bham.ac.uk/447/1/Martin09PhD.pdf
	https://en.wikipedia.org/wiki/Aging_movement_control
	https://en.wikipedia.org/wiki/Fine_motor_skill
	http://www.inclusivedesigntoolkit.com/betterdesign2/UCdex/dex.html
	http://www.dictionary.com/browse/dexterity
	http://themotorstory.org/building-strength-and.../finger-development-for-dexterity/
	http://jap.physiology.org/content/94/1/259

dx: dexterity.

each augmented query string. We gave the three sets of top ten documents (Table 4.1, 4.2) that Google returned (we retrieved the first two queries on June 1, 2016 and the third query on Feb 20, 2017) to the 30 evaluators, and asked them to hand pick the top five ranked articles per topic. The selections were based on relevance, comprehensiveness, and helpfulness in the domain of gerontology. We chose to have a small number of documents to be evaluated in an attempt to minimize human errors (false positives or false negatives).

With the 30 graders' top five ranked articles, we were able to obtain the graded ranking order of the ten articles per each augmented query.

We first assigned some linearly-increasing weights to the top five ranks respectively (see Table 4.3). Then we computed the total score of each article among the

Table 4.3: Linear Rank Weights

Rank No.	1	2	3	4	5
Weight	0.5	0.4	0.3	0.2	0.1

ten returned articles by adding up the products of the number of votes and the corresponding linear weight when the articles were chosen for any place in the top five. In detail, we computed the total score of each document as:

$$w_j = \sum_{i=1}^{5} (\text{No. of votes}) \times weight_i, j \in [1, 10].$$

We lastly sorted the total score of the ten documents in descending order to obtain the evaluators' graded ranking order.

4.2.2 Evaluation Methodology and Experiment Results

The 30 graders returned a total of 35 answer sheets for the three queries. Among the 35 submissions, there were ten valid entries for "calorie restriction," 14 for "fall prevention," and eight for "dexterity," while seven entries were invalid. Table 4.4 shows the computation results of our re-ranking algorithm for the top ten retrieved documents of each query. Table 4.5 displays the new ranking results from the evaluators and our re-ranking algorithm.

We used the Document Generality (DG) re-ranking algorithm (Yan, Li, and Song, 2006) as our experiment baseline. The algorithm follows a similar trail as ours, in that it extracts matching ontology terms from text, and uses ontology structure as

Table 4.4: Re-Ranking Scores

InitRank	f-p	c-r	dx
1	1.11185282741	0.97949153524	0.982258798424
2	0.832905887174	1.00597922231	0.832905887174
3	0.821198150665	0.94581726568	0.861589664529
4	0.947279042998	0.879452495662	0.977483471286
5	0.796779220466	0.796779220466	0.841246303145
6	0.883024207382	0.784583333358	0.803428056725
7	0.933640309711	0.937930424779	0.848699847134
8	0	0.873122550855	0.778999540533
9	0.843317600717	0.813930326999	0.820624694494
10	0.839634438398	0.771910780312	0.774130886433

initRank: Google rank order; f-p: fall prevention; c-r: calorie reduction; dx: dexterity.

Note: When we conducted the survey, document number 8 with "fall prevention" was no longer available online. We crossed it out and did not count it for evaluation.

Table 4.5: Ranking Results

f-p	GradedRank	4	5	10	1	3	7	6	9	2	[8]
	ReRank	1	4	7	6	9	10	2	3	5	[8]
	DGRank	1	7	10	4	9	6	2	3	5	[8]
c-r	GradedRank	1	2	6	8	7	9	3	10	5	4
	ReRank	2	1	3	7	4	8	9	5	6	10
	DGRank	7	2	1	8	3	9	4	10	5	6
dx	GradedRank	1	4	6	10	9	5	7	3	8	2
	ReRank	1	4	3	7	5	2	9	6	8	10
	DGRank	4	1	5	3	7	10	9	6	8	2

f-p: fall prevention; c-r: calorie reduction; dx: dexterity.

GradedRank: Rank order by evaluators.

ReRank: Rank order by our re-ranking algorithm.

DGRank: Rank order by Document Generality re-ranking algorithm.

the the oracle to spin out a more domain-relevant rank order. The result is presented in Table 4.6.

To compare the ranking results, we adopted popular ranking evaluation methodologies such as Pairwise Error Probability (e.g.: Kendall Tau Distance), Root Mean Square Error/Cosine Similarity, Discounted Cumulative Gain (DCG), and Mean Squared Error (MSE).

The Discounted Cumulative Gain (DCG) at a particular rank position p is calcu-

lated as the following:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log(i+1)},$$

where rel_i is the user graded relevance of the result at position i. The logarithmic reduction is to penalize the lower rank in the research result.

For easy calculation and maintain the result into a small range, we used the inverted rank as the rank order before plugging it into the formula.

For example, for rank order $\{1, 2, 3, 4, 5\}$, we use $\{1, 1/2, 1/3, 1/4, 1/5\}$ instead.

Table 4.6: Evaluation Results

		DCG	COSINE	KT	MSE
f-p	ReRank	**1.2521**	**0.1099**	0.4444	**1.0298**
	DGRank	1.0975	0.0899	**0.4722**	1.0380
	Google	1.0897	0.0879	0.4722	1.0392
c-r	ReRank	**1.4007**	**0.1228**	**0.3555**	**1.0147**
	DGRank	1.1520	0.0950	0.3111	1.0292
	Google	1.6147	0.1488	0.4667	1.0262
dx	ReRank	**1.5545**	**0.1568**	**0.4222**	1.0228
	DGRank	1.4087	0.1355	0.2888	**1.0161**
	Google	1.4683	0.1415	0.6222	1.0305

f-p: fall prevention; c-r: calorie reduction; dx: dexterity.
DGRank: Document Generality re-ranking algorithm.
ReRank: Our re-ranking algorithm. Google: Google base rank.

4.2.3 Experiment Result Analysis

We set the linear combination coefficient of our re-ranking formula to 0.85 due to a conservative attitude, because we want to respect the initial search order. However, Table 4.7 shows a strong correlation between documents and the ontology dictionary terms. One of the documents under query "Fall Prevention" contained 27 matched terms in its textual content. Among the total thirty documents, only two documents did not return any term match. The coefficient therefore could be adjusted more in the re-ranker's favor.

From the results of our experiments (Table 4.6), we could see that DG algorithm only outperformed our re-ranking algorithm in two ((f-p:KT and dx:MSE) out of the twelve experimental categories; our algorithm was superior to or equally good for the rest. It showed a promising performance of our re-ranking algorithm. Another observation was made through comparing our new rank with Google rank order. Though we did not treat Google initial rank as one of our baselines, we did calculate its performance as a reference point. Our re-ranker did better on "Fall prevention" and "dexterity" among the three queries over Google, which nonetheless purports levels of viability and validity of our re-ranking algorithm.

Table 4.7: Number of Matched (Ontology) Terms Found in Each Retrieved Document (In reference to Table 4.1, 4.2)

initRank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
f-p	27	0	5	13	12	5	11	NULL	6	12
c-r	9	13	7	4	5	13	15	9	6	2
dx	9	0	1	15	5	2	4	1	3	5

initRank: Google rank order.

f-p: fall prevention; c-r: calorie reduction; dx: dexterity.

Note: When we conducted the survey, document number 8 with "fall prevention" was no longer available online.

Chapter 5

THE ONTOLOGY LEARNING MODELS

This chapter was structured based on (Zhao and Zhang, 2018).

5.1 The Ontology Hierarchy Extension Learning Models

We proposed two learning models of grafting some new node t_{n} in G to support decision making , using supervised learning with Neural Networks and Naïve-Bayes.

5.1.1 New Concept Extraction – Unsupervised Learning

Using unsupervised learning techniques, we can train the system to recognize new terms (not yet in ontology vocabulary), that frequently (at least $\tau \geq 1$ occurrences) show up in the metadata (keywords, description, title, etc) fields of the top k ranked articles per reference node t_{ref} .

We first performed a new term mining by looping through the metadata of the documents returned from a web search, via some general-purposed authoritative search engines, per a query term that corresponds to an existing vocabulary term t_{ref} in the ontology dictionary, which in turn corresponded to a node on the ontology graph.

To mine the nouns and phrases from metadata, we will need NLP techniques. There are intensive studies relating to this topic and many well-tested algorithms and tools available.

More specifically, new concept extraction is done by training the system to recognize nouns and phrases that are not yet in ontology vocabulary, shown up in the metadata fields (keywords, description, title, etc) of the top k ranked articles per reference term t_{ref}, using NLP techniques (Python textblob). We call the set of nouns and phrases, the *candidate terms*. We then count each *candidate term*'s frequency, namely, in how many documents the term is present. The most frequented term or terms (frequencies is greater than a threshold τ, $\tau > 1$) as the *new terms* will be the *new terms* we want to analyze and add to the ontology dictionary and and the graph.

Following the above steps, we use the traditional TF/IDF (term frequency inverse document frequency) weighing method to sift through each *candidate term* against all documents. Only one term among the candidate terms whose frequency is greater than a threshold τ and $\tau \geq 1$ is to be picked at a time as the new term of interest, t_{n}.

This is a step relative to Domain Terminology Extraction, that the resulting list of terms has to be filtered by a domain expert in practice. We therefore hand pick or randomly select one term, t_{n}, from the candidate list.

New term t_{n} was the prior condition for our ontology hierarchy extension algorithms.

Once we got t_{n}, we either performed a web data search on the new term to see if there was any relationship between t_{n} and all existing ontology terms. Or, we performed a web data search on t_{ref}'s neighboring nodes to see if any relationship existed between the neighboring clusters and t_{n}.

5.1.2 WAY Neural Networks Learning Model

In this approach, we performed an augmented search with a domain characteristic classifier (DCC) (Zhao and Zhang, 2017) over t_{n} and collected the k top ranked documents. We then parsed the textual content of the k documents and got a tally

of each matched ontology term, to form a distribution. We threw both the centroid of the distribution and the local/global maximum of the distribution into a Neural Networks classifier, with which we made our best bet to place t_{n} either in the child or the sibling position of t_{ref} in G.

The metaphoric WAY ("We Are in You") uses "we" to indicate the seed ontology terms, and "you" to represent t_{n}.

The algorithm features the following steps:

1. Performing an augmented search over query string t_{n} on the web using some authoritative search engines, and fetching the top k documents.

2. Parsing through each retrieved document (text, metadata) of the k documents and collecting a distribution of matched ontology terms ($\mathbf{t}_{t_{\mathrm{n}}}$) and their corresponding frequency (number of documents) ($\mathbf{f}_{t_{\mathrm{n}}}$) .

3. Computing the *centroid*, D_{cent}, the geometric center of the distribution.

   We first calculated the distance from rN to each term $t \in \mathbf{t}_{t_{\mathrm{n}}}$. The frequency vector $\mathbf{f}_{t_{\mathrm{n}}}$ is then used to assign weights to each data point in $\mathbf{t}_{t_{\mathrm{n}}}$.

D_{cent} is defined as:

$$D_{\text{cent}} = \frac{\sum_{t \in \mathbf{t}_{t_{\text{n}}}, f \in \mathbf{f}_{t_{\text{n}}}} d(t, rN) \times f}{\sum_{f \in \mathbf{f}_{t_{\text{n}}}} f}, \tag{5.1}$$

where $d(t, rN)$ is the shortest path from node t to rN and f is the frequency count of the corresponding t.

Let d_{ref} be the distance from t_{ref} to rN.

To reduce data noise and computation complexity, we processed only some nodes t in G, whose distance $d(t, rN)$ is in the interval $[d_{\text{ref}} - \delta 1, d_{\text{ref}} + \delta 2]$, where $0 < \delta 1, \delta 2 < Depth(G)$. For example, we set $\delta 1 = 1$ and $\delta 2 = 2$. The range therefore is $[d_{\text{ref}} - 1, d_{\text{ref}} + 2]$. If $d_{\text{ref}} = 3$, we will process some nodes t, that $2 \leq d(t, rN) \leq 5$.

4. Finding the local maximum (the term with the highest frequency), or the average of local maxima (terms with the same highest frequency), $D_{\max}$.

$$D_{\max} = \frac{\sum_{v \in V_{\max}} d(v, rN)}{|V_{\max}|}, \tag{5.2}$$

where $V_{\max}$ is the set of local maxima. $|V_{\max}|$ is the total number of elements in $V_{\max}$.

We use D_{cent}, $D_{\max}$, and d_{ref} as the neurons to feed our neural networks com-

putation system.

5. Computing the average linkage D_{avg}.

   The *average linkage* is a term we borrowed from Hierarchical Agglomerative Clustering (HAC). Here it means a linear combination value of D_{cent} and D_{max}.

$$D_{\text{avg}} = \alpha \times D_{\text{cent}} + (1 - \alpha) \times D_{\text{max}}, \tag{5.3}$$

   where $\alpha = [0, 1]$ is an adjusting factor. We set $\alpha = 0.85$, an estimated probability value that the centroid value is closer to the ground truth than the local maximum value.

   (The reason we performed this step was because the experiment results using three neurons: D_{max}, D_{cent}, and d_{ref} (the shortest distance from rN to t_{ref}), were not ideal. This was mainly due to the strong correlation between D_{max} and D_{cent}. The necessary independence among neurons was suggested by the experiments.)

6. Classification using neural networks model. We used Sigmoid activation function for our neural networks learning algorithm.

Pre-activation function:

$$f_{pre} = D_{\text{avg}} \times w1 + d_{ref} \times w2 + b,$$

where $w1, w2$ are the weights and b is the bias, d_{ref} is the shortest distance from rN to t_{ref} .

Activation function (Sigmoid function — Logistic Function):

$$\sigma(z) \equiv 1/(1 + e^z).$$

Combining the above two formulae, we have our neural networks prediction function, *pred*:

$$pred = \sigma(f_{\text{pre}}).$$

7. Computing t'_{n}.

We use label 0 and 1 to represent class SIBLING and CHILD, respectively:

$$t'_{\text{n}} = \begin{cases} 0, \text{ if } pred \leq 0.5, \\ 1, \text{ otherwise.} \end{cases} \tag{5.4}$$

5.1.3 YAU Naïve-Bayes Learning

The second approach we proposed was to perform an augmented search over t_{ref}'s sibling nodes and child nodes and to collect the top k documents per node search. We then looked for t_{n} in each returned document and kept the frequency counts. We performed a classification, using Naïve-Bayes, based on the weights collected from the two sets of nodes, and decided which class label, SIBLING or CHILD, to be tagged to t_{n}.

In this approach – YAU ("You Are in Us"), we searched t_{n} ("you") in the documents returned by some authoritative search engines through querying some seed ontology terms ("us").

The Naïve-Bayes model is fast, highly scalable, and great for binary classification. It assumes that the probability of each input attribute belonging to a given class is independent of all other attributes, which is what we surmised in this case.

1. Defining prior probability

   Let $\mathbf{sib_i}$, $\mathbf{child_i}$ be the sibling node vector and child node vector of t_{ref}.

   We denote the Prior Probabilities of two classes, SIBLING and CHILD, as $P(sibling)$ and $P(child)$. $P(sibling)$ was approximated as the number of all

non-leaf nodes in G over twice the total nodes in G, and $P(child)$ is the total non-root nodes over twice the total nodes in G. Thus, $P(child) > P(sibling)$.

$$P(sibling) = \frac{|V| - |V_{\text{leaf}}|}{2 \times |V|}, \quad Depth(G) \geq 2. \tag{5.5}$$

$$P(child) = \frac{|V| - 1}{2 \times |V|}, \tag{5.6}$$

where $Depth(G) \geq 2$ ensures a non-zero value for $P(sibling)$.

Now we will detail the steps of calculating the value of $\mathbf{sib_i}$ and $\mathbf{child_i}$ using Bayes theorem in the following.

2. Performing an augmented search over each term in $\mathbf{sib_i}$ and $\mathbf{child_i}$, and fetching the top k documents per term.

3. Parsing through each retrieved document of the augmented query term in $\mathbf{sib_i}$ and $\mathbf{child_i}$, and counting the number of documents, $N(t_{\text{n}}) \leq k$, in which the new term t_{n} is found. $N(t_{\text{ref}})$ is equivalent to k, representing the number of documents containing the reference term t_{ref}.

   In the case that t_{n} is a composite word or a phrase and there is no match in k documents, we would perform a bag-of-words search, namely word by word search, instead.

4. Calculating Bayes value of each term in $\mathbf{sib_i}$ and $\mathbf{child_i}$.

The Likelihood of being in the SIBLING or CHILD class for t_{n} is computed as:

$$P(t_{\mathrm{n}}|C) = \frac{N_C(t_{\mathrm{n}}) + \alpha}{N_C(t_{\mathrm{n}}) + \alpha \times N_C(t_{\mathrm{ref}})}, \tag{5.7}$$

where C is the class label which is either SIBLING or CHILD. The smoothing priors $\alpha \geq 0$ accounts for features not present in the search space and prevents zero probabilities in further computations. We use Laplace smoothing and set $\alpha = 1$.

Thus, the equation can be simplified as:

$$P(t_{\mathrm{n}}|C) = \frac{N_C(t_{\mathrm{n}}) + 1}{N_C(t_{\mathrm{n}}) + N_C(t_{\mathrm{ref}})}. \tag{5.8}$$

In case that $\mathbf{sib_i}$ or $\mathbf{child_i}$ is empty, we define a default likelihood value as the 30% of the maximum likelihood value. The maximum value is when $N_C(t_{\mathrm{n}}) = N_C(t_{re})$, thus:

$$P(t_{\mathrm{n}}|C)_{max} = \frac{N_C(t_{\mathrm{n}}) + 1}{N_C(t_{\mathrm{n}}) + N_C(t_{\mathrm{ref}})} \approx \frac{N_C(t_{\mathrm{n}})}{2 \times N_C(t_{\mathrm{n}})} = 0.5. \tag{5.9}$$

Therefore, we have $0.5 \times 30\% = 0.15$ as the default likelihood value of the class if its related vector is empty.

The Predictor Prior Probability or Evidence is:

$$P(t_{\mathrm{n}}) = N(t_{\mathrm{n}})/N(t_{\mathrm{ref}}), \tag{5.10}$$

where $N(t_{\mathrm{n}}) = N_s(t_{\mathrm{n}}) + N_c(t_{\mathrm{n}})$ is the combined number of documents that contain the query term t_{n}, and $N(t_{\mathrm{ref}})$ is the total number of documents processed.

Finally, we compute the Posterior Probability:

$$P(C|t_{\mathrm{n}}) = \frac{P(C)P(t_{\mathrm{n}}|C)}{P(t_{\mathrm{n}})}. \tag{5.11}$$

5. Computing t'_{n}.

We use label 0 and 1 to represent class SIBLING and CHILD, respectively.

$$t'_{\mathrm{n}} = \begin{cases} 0, \text{ if } P(sibling|t_{\mathrm{n}}) \geq P(child|t_{\mathrm{n}}), \\ 1, \text{ otherwise.} \end{cases} \tag{5.12}$$

5.1.4 Validation and Verification

A validation and verification process is necessary after we make a prediction of the ontology hierarchy extension. We proposed two semi-automation methods for validation and verification: 1) Using a user feedback/voting system to evaluate t'_{n}. 2) Using usage data analysis and human experts to perform checks and verification.

5.2 Experiments

5.2.1 Experiment Setup and Datasets

Our experimental system was developed using Python, PHP, and Java. It used the existing Plant Ontology [1] and Human Disease Ontology [2] developed by OBO Foundry [3] as the supervised learning ontology domains and data source. Both ontologies were written in Open Biomedical Ontologies (OBO) language. We randomly picked a total of 34 data points (Table(s) 5.1 and 5.2) from the above two ontologies as our training, validation and test data sets for our algorithms. There were 16 data points with CHILD relationship, and another 18 with SIBLING relationship, which ensured a relatively balanced data bed.

For notation convenience, we used labels 0 and 1 to represent class SIBLING and CHILD respectively. The notation was carried on throughout the EXPERIMENTS section.

Our ontology structures may have been slightly different than the original ones, since we have the restriction that one term can only reside in one nodal place in the ontology tree, while the original ontologies may have terms that appear in multiple locations.

[1] `http://www.obofoundry.org/ontology/po.html`

[2] `http://obofoundry.org/ontology/doid.html`. The ontology asserts *IS-A* hierarchy.

[3] The Open Biomedical Ontologies (OBO) Foundry (now The Open Biological and Biomedical Ontologies (OBO) Foundry) is a collaborative experiment involving developers of science-based ontologies. `http://www.obofoundry.org`.

5.2.2 WAY Neural Networks - Data Training and Validation

We first experimented with backpropagation algorithm to the training sets. The training sets quickly reached to almost 100% accuracy rate, but performed poorly in the test sets. We then switched to feedforward propagation.

We computed the 34 data points using our WAY algorithm. Each data point contained three elements: D_{avg}, d_{ref}, and ground truth. The following dataset depicts the format.

```
Training/Validation Dataset
[3.32903225806, 2, 1],[3.64312977099, 3, 1],[4.67342105263, 4, 1],
[3.37862694301, 3, 0],[3.91593406593, 3, 1],[3.98868613139, 3, 0],
[3.57453703704, 3, 0],[4.13131868132, 3, 1],[4.17284946237, 3, 0],
[3.5119140625, 2, 1],[3.5119140625, 2, 1],[3.45490797546, 3, 0],
[4.15867346939, 3, 1],[4.58667582418, 4, 0],[3.5036, 2, 1],
[2.5772972973, 1, 0],[4.58908839779, 4, 0],[4.01358024691, 3, 0],
[4.01358024691, 3, 0],[2.7884057971, 2, 1],[3.3345, 3, 0],
[4.09965517241, 4, 0],[3.80073260073, 3, 1],[4.06801152738, 4, 0]
Testing
[3.70743670886, 3, 1],[4.08445512821, 4, 0],[4.27575376884, 4, 1],
[2.1375,2,0],[2.59583333333,2,1],[2.60088174274, 2, 1],
[3.10143678161,3,0],[3.28541666667,3,0],[2.18259911894,2,1],
[3.54982517483, 3, 0]
```

We used the Least-Squares cost function as our loss function, i.e., minimizing the squares of the differences between targeted values and values predicted by the model.

$$J(\theta) = \frac{1}{2N} \sum_{i=1}^{N} (h_\theta(x^{(i)}) - y^{(i)})^2,$$

where $y^{(i)}$ is the targeted value of the data set for the ith data point $x^{(i)}$. h_θ is the model function. θ refers to all the adjustable parameters ($w1, w2$, and b in the model function) and x is the vector of input variables, namely D_{avg} and d_{ref}.

We used the average error for the training dataset. We then ran gradient descent on this error function to minimize the cost. We took the partial derivative of the cost function, $\frac{\partial}{\partial\theta_j}J(\theta_j)$, and computed the best values for $w1, w2$, and b.

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_j),\ \forall\,\theta_j \in \{w1, w2, b\},$$

where $0.001 \leq \alpha \leq 10$ is the learning rate. We use the assignment notation $:=$ to denote the batch gradient descent update rule.

We now describe our Gradient Descent hyperparameters (initialization):

- Learning rate α: 0.01.
- Number of training iterations: 10,000.
- Parameter initialization: we used random numbers to initialize $w1, w2$, and b.
- Input data: 24 data points.

We obtained the optimal learning rate from a preliminary iteration of all the test data via the above configuration. We then performed the k-fold cross validation supervised learning process.

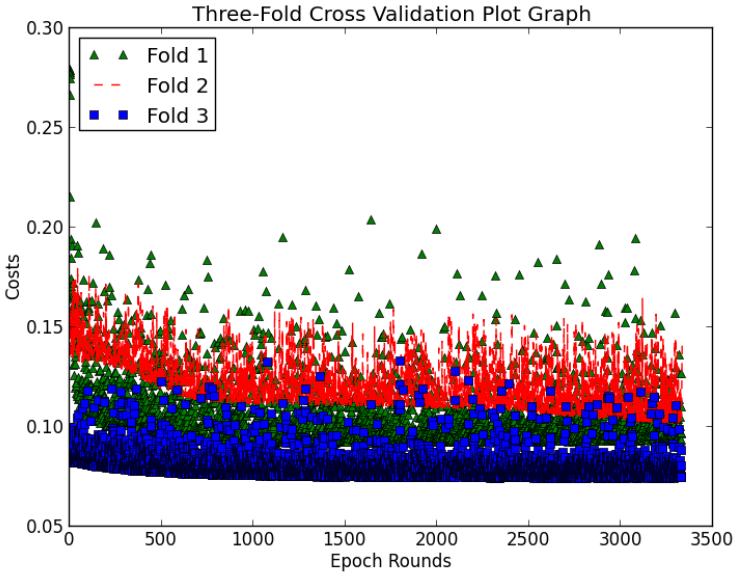

Figure 5.1: WAY Neural Networks 3-Fold Cross Validation Training Data Graph (all data points)

The system used, in particular, 3-fold cross validation (Fig. 5.2). Parameters set for the training and validation were:

- Learning rate: 0.13
- Parameter Initialization: we use random numbers to initialize $w1, w2$, and b.
- Input data: 24 data points (16 for training, 8 for validation).
- Number of sample data iterations: $50,000$.
- Maximum number of epochs: 100.
- Accuracy rate threshold: $\tau = 0.7$.
- For every 15 records, we averaged the error rates, to avoid overfitting.

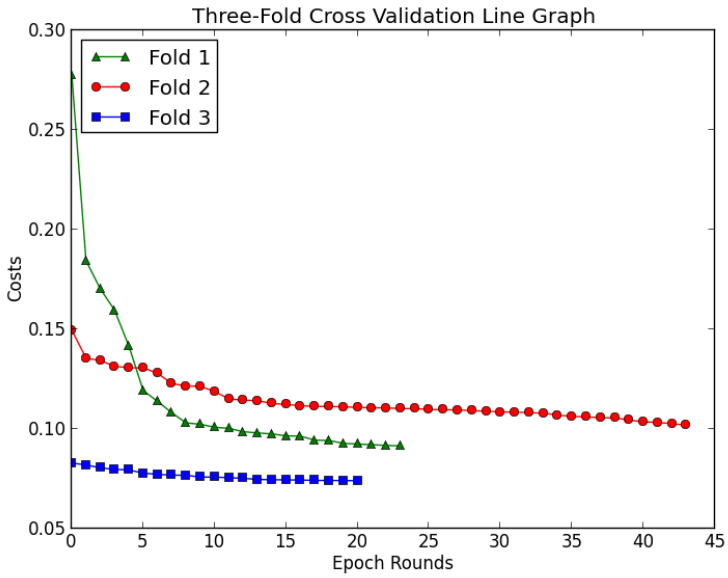

Figure 5.2: WAY Neural Networks 3-Fold Cross Validation Training Data Graph (selected data points)

From the above 3-fold training and validation process, we obtained:

- $w1 = 15, 2.8089129890331557$,
- $w2 = -7.9431967425717467$, and
- $b = 13.633143894205276$.

We then plugged the data in to evaluate our test set. Table 5.3 shows the results, which indicate an accuracy rate of 0.8.

5.2.3 YAU Naïve-Bayes - Data Training and Validation

We collected $P(sibling|t_{\mathrm{n}})$, $P(child|t_{\mathrm{n}})$, and the ground truth (SIBLING – 0/CHILD – 1) for each data point of our 34 data records. Using the raw Bayes classification, we got eight out of ten test data correct, as illustrated in Table 5.4.

described in the following list.

```
 Training/Validation Dataset
[0.345108354271,0.2247,1],[0.354730564169,0.445860533253,1],
[0.332890506171,0.1665,1],[0.282956930245,0.386120945637,0],
[0.309072831569,0.14865,1],[0.635387463634,0.354596756699,0],
[0.530109564215,0.68115534644,0],[0.328635003413,0.321758947957,1],
[0.322347807489,0.1682904,0],[0.260643617969,0.99710383981,1],
[0.317820775431,0.323567118379,1],[0.350408392376,0.362088497727,0],
[0.319640782267,0.362325539456,1],[0.347047738693,0.18738,0],
[0.272959817165,0.308746610294,1],[0.278013092723,0.293607031833,0],
[0.347047738693,0.062460741206,0],[0.290596850093,0.307332334632,0],
[0.30923356816,0.301051685471,1],[0.286846173422,0.643334618418,1],
[0.472949693013,0.253817137117,0],[0.320100502513,0.15,0],
[0.30236263564,0.361692411704,1],[0.391261118426,0.2724,0]
Test Dataset
[0.348577512887, 0.462672157082, 1], [0.280822634427, 0.331659696755, 0],
[0.0694061318783, 3.17997578565, 1],[0.147299472416, 1.27028445213, 1],
[0.218221764631, 0.289322577766, 0],[0.204648915138, 0.314255415536, 1],
[0.205522999791, 0.09834, 0],[0.273410110156, 0.22686, 0],
[0.21660763704, 0.328835428819, 1], [0.212739540334, 0.10344, 0]
```

We also experimented using Gaussian classifier with Bayes to further calculate the properties to predict t'_{n}. The steps were as follows:

1. Constructing Gaussian classifier.

After getting each element's Bayes value, we took the mean μ_C, and the standard deviation σ_C of the training data, where C denoted one of the two classes, SIBLING and CHILD.

We then calculated the Gaussian probability gau_C:

$$\mu_C = \frac{1}{N}\sum_{i=1}^{N} P(C|t_{\mathrm{n}})_i.$$

$$\sigma_C = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(P(C|tn)_i - \mu_C)^2}.$$

$$gau_C = P(x|C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} e^{(-\frac{(x-\mu_C)^2}{2\sigma_C^2})},$$

where x is the input value, and σ^2 is the variance.

2. Computing t'_{n}.

Let gau_0 denote the Gaussian probability for class SIBLING, gau_1 for CHILD.

$$t'_{\mathrm{n}} = \begin{cases} 0, \text{ if } gau_0 \geq gau_1 \ , \\ 1, \text{ otherwise.} \end{cases} \tag{5.13}$$

By running the above tasks, we achieved an accuracy rate of 0.8 as well. The following log data depicts the above tasks' results:

```
('Split dataset for 34 records: training 24 records, test 10 records) ('
```

```
SUMMARY: ', {0: [(0.58680166925396759, 0.12473888115325908),
(0.41319833074603252, 0.12473888115325908)], 1: [(0.48391380239468401,
0.13355541307795385), (0.51608619760531593, 0.13355541307795382)]}) (1,
 'correct ')(2, 'incorrect ')(3, 'correct ')(4, 'correct ')(5, '
incorrect ')(6, 'correct ')(7, 'correct ')(8, 'correct ')(9, 'correct
')(10, 'correct ') ('Accuracy: 80.0)
```

Additional Test Set

We added an additional test set after our first round of experiments to further test and validate our models. The test set includes 10 additional data points from Plant Ontology (Table 5.5).

The accuracy rate for the test result of the additional ten records was 0.7 and 0.8 in the WAY Neural Networks approach and YAU Naïve Bayes approach, respectively.

5.2.4 Evaluation Methodology

Since our data set was balanced in terms of the distribution of the two classes (SIBLING and CHILD), we used Accuracy (ACC), Matthews Correlation Coefficient (MCC), Sensitivity and Specificity to evaluate our experiments' results. Table 5.6 describes the shorthand notations used in the ensuing methodology definitions.

ACC is defined as

$$ACC = \frac{\sum TP + \sum TN}{\sum TotalPopulation}.$$

MCC is in essence a correlation coefficient between the observed and predicted binary classifications; it returns a value between -1 and $+1$. A coefficient of $+1$ represents a perfect prediction, 0 no better than random prediction and -1 indicates total disagreement between prediction and observation [4].

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.$$

Sensitivity or true positive rate (TPR), equivalent with hit rate, is defined as:

$$TPR = TP/(TP + FN).$$

Specificity (SPC) or true negative rate is defined as:

$$SPC = TN/(TN + FP).$$

5.2.5 Experiment Analysis

Our first round of experimental results showed that WAY Neural Networks approach and YAU Naïve-Bayes approach achieved the same accuracy rate. The Neural Networks approach may have shown better performance in MCC, TPR, and SPC over the Bayes approach. The plain Naïve-Bayes approach and Naïve-Bayes with Gaussian classifier were identical in results and performance, which indicates that Gaussian probability may not have obvious advantages to add to the Bayes classifier.

[4] `https://en.wikipedia.org/wiki/Matthews_correlation_coefficient`

The FP and FN records of the two approaches, WAY and YAU, were different, which signifies that the two approaches may have captured different hidden features of the ontology data and structural relationship.

We observed that the two FP records in the Bayes approach have values of the two classes in a very close range. This may imply the semantic soundness of the approach.

The additional test dataset, our second round of experiments with an 0.8 ACC in YAU method and 0.7 in WAY, further proved the efficiency of YAU approach, but was less supportive of WAY model. Nonetheless, an accuracy rate of 0.7 was still within a supportive range.

Our experiments were conducted over a relatively small dataset but using the industry trusted ontologies. The results nonetheless affirmed our hypothesis and the promising nature of our learning algorithms.

Table 5.1: Training Dataset (Ontology: Plant)

	Reference Term (t_{ref})	**New Term** (t_{n})	**Class** (relationship)
1	phyllome sinus	leaf sinus	1
2	arilloid	aril	1
3	phyllome substomatal cavity	leaf substomatal cavity	1
4	sporangium locule	fruit locule	0
5	sporangium locule	anther locule	1
6	leaf mesophyll intercellular space	substomatal cavity	0
7	petiole canal	hilum groove	0
8	petiole canal	abaxial petiole canal	1
9	hilum groove	petiole canal	0
10	axil	bract axil	1
11	substomatal cavity	leaf mesophyll intercellular space	0
12	petiole canal	resin canal	0
13	petiole canal	abaxial petiole canal	1
14	abaxial petiole canal	adaxial petiole canal	0
15	collective phyllome structure development stage	androecium development stage	1
16	collective plant organ structure development stage	multi-tissue plant structure development stage	0
17	abaxial petiole canal	adaxial petiole canal	0
18	cultured plant cell	native plant cell	0
19	cultured plant cell	plant protoplast	1
20	trichome	glandular trichome	1
21	multicellular trichome	glandular trichome	0
22	archegonium megagametophyte	embryo sac	0
23	megagametophyte	embryo sac	1
24	embryo sac	archegonium megagametophyte	0

Table 5.2: Test Dataset (Ontologies: Plant and Human Disease)

	Reference Term (t_{ref})	**New Term** (t_{n})	**Class** (relationship)
	`Plant Ontology`		
25	megagametophyte	archegonium megagametophyte	1
26	egg apparatus	embryo sac central cell	0
27	egg apparatus	synergid	1
	`Disease Ontology`		
28	chondrodysplasia punctata	chromosomal disease	0
29	chondrodysplasia punctata	rhizomelic chondrodysplasia punctata	1
30	chromosomal disease	Angelman syndrome	1
31	Angelman syndrome	Down syndrome	0
32	Edwards syndrome	egg apparatus	0
33	ciliopathy	Joubert syndrome	1
34	Meckel syndrome	Joubert syndrome	0

Table 5.3: WAY Neural Networks Test Results (Ontologies: Plant and Human Disease)

	pred	t'_{n}	GT	T|F
25	0.55414208353552363	1	1	T
26	0.0012708926971506554	0	0	T
27	0.0021731050261987062	0	1	F
28	0.97704062590619112	1	0	F
29	0.99355650843227683	1	1	T
30	0.99364665941563679	1	1	T
31	0.18470647279465863	0	0	T
32	0.27527996563740792	0	0	T
33	0.97971701523019628	1	1	T
34	0.44391296751654125	0	0	T

GT: Ground truth. T|F: True or False.

Table 5.4: YAU Raw Bayes Classifier Test Results (Ontologies: Plant and Human Disease)

	$P(sibling\|t_n)$	$P(child\|t_n)$	t'_n	GT	T|F
25	0.348577512887	0.462672157082	1	1	T
26	0.280822634427	0.331659696755	1	0	F
27	0.0694061318783	3.17997578565	1	1	T
28	0.147299472416	1.27028445213	1	1	T
29	0.218221764631	0.289322577766	1	0	F
30	0.204648915138	0.314255415536	1	1	T
31	0.205522999791	0.09834	0	0	T
32	0.273410110156	0.22686	0	0	T
33	0.21660763704	0.328835428819	1	1	T
34	0.212739540334	0.10344	0	0	T

GT: Ground truth. T|F: True or False.

Table 5.5: Test Dataset II (Ontology: Plant)

	Reference Term (t_{ref})	**New Term** (t_n)	**Class**
35	plant anatomical space	anther pore	1
36	axil	anatomical space	0
37	branch axil	bract axil	0
38	axil	leaf axil	1
39	resin canal	petiole canal	0
40	intercellular space	substomatal cavity	1
41	substomatal cavity	phyllome substomatal cavity	1
42	phyllome substomatal cavity	leaf substomatal cavity	1
43	phyllome stomatal pore	adjacent_to substomatal cavity	0
44	resin canal	archegonium neck canal	0

Table 5.6: Shorthand Notation Lookup Table

TP	True Positive, ground truth and test result are both of label 1.
TN	True Negative, ground truth and test result are both of label 0.
FP	False Positive, ground truth is label 0, but test result is label 1.
FN	False Negative, ground truth is label 1, but test result is label 0.

Table 5.7: Experiment Evaluation

	Accuracy	MCC	TPR	SPC
NN	0.8	0.6	0.8	0.8
RB	0.8	0.4286	1	0.6
BG	0.8	0.4286	1	0.6

NN: WAY - Neural Networks
RB: YAU - Raw Bayes classifier
BG: YAU - Bayes with Gaussian probability

Chapter 6

THE DLKS FRAMEWORK

6.1 Overview

We call our knowledge search framework the Dynamic Lightweight Knowledge Search (DLKS – can be pronounced as "dee-links").

MITRE [1] defines an architecture framework as "an encapsulation of a minimum set of practices and requirements for artifacts that describe a system's architecture." We have already laid out the core technical building blocks for our knowledge search framework. Now we are ready to give the basic structure underlying the framework.

The framework will be an abstract architectural framework. It will not specifically address any level of underlying implementation details such as platforms or technolo-

[1] `https://www.mitre.org`

gies in use. The practice accords with separation of concerns (SoC [2]), a principle used in software development. In this case, we separate the theoretical and architectural specifications from the implementation practice, to make the system more generic, dynamic, and adaptable.

The framework focus on the characteristics of being a "light-weighted" system.

- Each of the domain ontologies is of minimum number of nodes, since the IS-A relationship restricts the level of extension of sub-concepts, if the domain knowledge is within a reasonable scope. The assumption is that users in general will not search those extreme general concepts such as "thing," "animal," or "computer." But they may search things like "artificial intelligence," "Apple computers," or "Victorian art." In addition, for each domain search query, the system only fetches a neighborhood of sub-concept (IS-A) or related concepts (HAS-A) of d-distance to the searched domain concept.

- The database storage needed for the system will be lightweight. We only need to store document URLs (with or without blurbs), metadata, logs, and most importantly, the ontologies. The necessary database storage is mainly for caching purpose to enable a faster search.

- Reasonable effort to refresh database data on a regular base. If the re-ranker engine is optimized, all searches can be done ad hoc. However, we may also refresh the ranking URLs offline on a regular schedule, say daily or bi-weekly. In

[2] In computer science, separation of concerns (SoC) is a design principle for separating a computer program into distinct sections, such that each section addresses a separate concern. A concern is a set of information that affects the code of a computer program (wiki).

practice, the above two will both be in operation, because there will be new domain knowledge requests, while the related ontologies are not in system. Hence, new ontologies will be created and the URL ranking and fetching will be done at run time.

We may leave the initial seed ontology creation to the users, so that they can start off with extremely simple ontologies. The system will help enhance and perfect the ontologies, along with other users who show interest in the domain. It follows a model of openly editable content and a collaborative platform, which has been proven to be a success by Wikipedia.

- A lightweight system scaling effort. Upon a large amount of requests, an ad hoc re-ranking effort using our full set of re-ranking algorithms should be avoided. Instead, we can use the lightweight version – using DCC-only re-ranking approach to perform the ad hoc re-ranking, but leave the in-depth re-ranking at an offline time. As a result, the efforts for scaling will be minimum. This is due to the massive efforts already undertaken by Google and other search engines. Our system is built on top of the giants.

6.1.1 Ontology Content Management System

The Ontology Content Management System (OCMS) is a core component of the backend system of DLKS framework. It provides a convenient integrated development environment (IDE) platform for users to create and update ontologies.

If under proper control, not only the system administrators but the web users as well, may utilize the OCMS for their domain ontology management.

Though we tend to give an abstract direction for the system to be built, we thought it is worth mentioning a great ontology building system – OBO (See Appendix A).

6.2 Logical Architecture

We sketch out the three-tier architecture of our DLKS framework with emphasis on logic tier but minimum description of the presentation tier.

Figure 6.1 presents a logical architecture diagram that illustrates the information flow and linkages between various modules and main processes. Any subsequent system architecture should be derivable from this logical framework.

6.2.1 Presentation Tier

This tier is mainly composed with two groups of web services: Search and Interface.

The Search Engine Web Service performs the tasks to communicate with generic search engine(s) and pass the query strings over, and fetch the research result sets

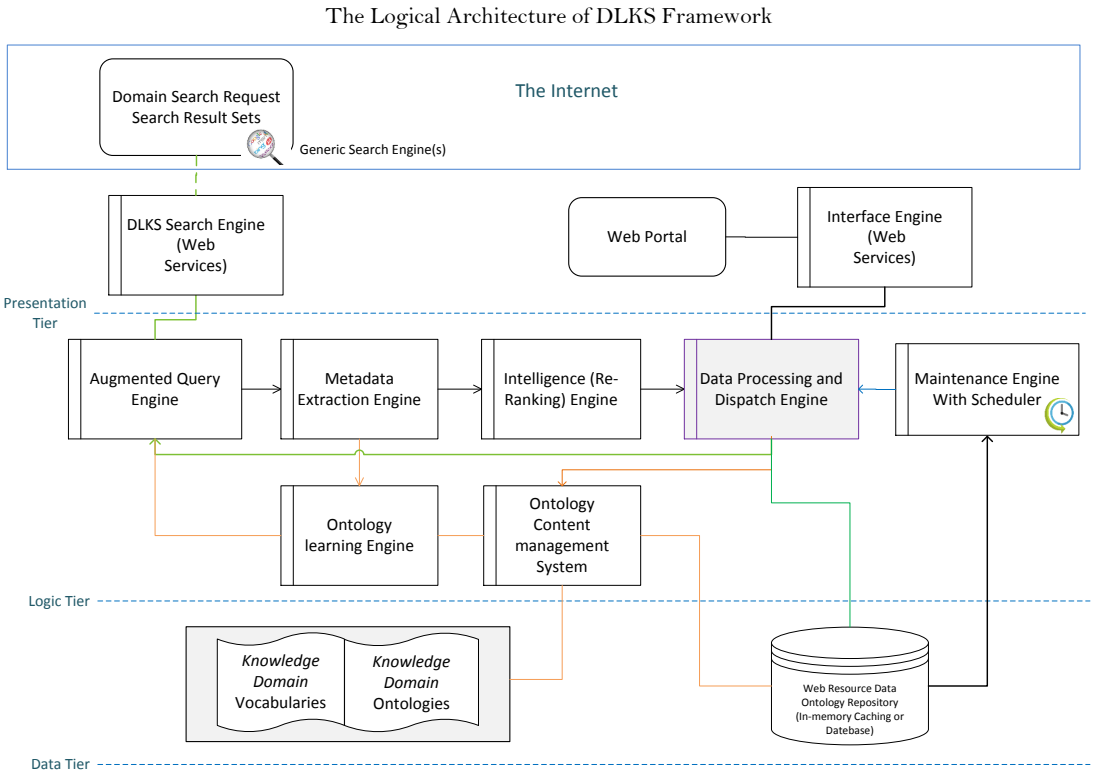

Figure 6.1: DLKS Framework Architecture Diagram

back to the application tier.

The Interface Engine Web Services act as the middle man between the front-end web portal and the application layer.

6.2.2 Logic Tier

Re-Ranking Route

1. The Data Processing and Dispatch Engine (DPDE) fetches search queries either from the Interface Engine or from the Maintenance Engine With Scheduler (MEWS). It then dispatch the queries to the Augmented Query Engine.

   If the query is from the Interface Engine, the DPDE will look up the database to see if there is an existing entry for related query. If so, the response set will be fetched directly from the database and sent back to the Interface Engine.

2. The Augmented Query Engine (AQE) composes and sends augmented query strings corresponding to nodes in the domain ontology tree. It communicates using APIs specified by DLKS Search Engine Web Services. It then takes the $n > 0$ search results back from DLKS Search Engine.

3. The Metadata Extraction Engine takes data from AQE and extracts the full text and metadata, such as title, description, keywords, authoring date, etc, out of every search result and produces DLKS required metadata for further processing.

   This step can be omitted if using the lightweight DCC-only augmented search.

4. The output of the above step subsequently feeds into the Re-Ranker Engine, which in turn produces top ranked domain-specific $k <= n$ results.

This step can be omitted if using the lightweight DCC-only augmented search.

5. Finally, we rely on DPDE to either store all data into the database or a caching repository with necessary indexing mechanism; or present the data to the Interface Engine for Web Portal to display; or do both.

Ontology Learning Route

1. The Ontology Content Management System (OCMS) gets request of ontology building or ontology hierarchy extension requests from DPDE.

   There are two cases of ontology learning request:

   - DPDE gets a request from Interface Engine, and forwards the request to OCMS
   - MEWS triggers ontology hierarchy extension update task and sends batch requests to DPDE, DPDE forwards the request to OCMS

   OCMS either sends ontology data to database or gives it to the Ontology Learning Engine (OLE) for further processing.

2. The OLE takes ontologies from OCMS, and then sends to AQE for search tasks. After Metadata Extraction Engine processes the search result set, the data is sent back to OLE. OLE updates the related ontology tree and sends updated ontology back to OCMS.

6.2.3 Data Tier

It is the storage and repository layer for all ontology-related data and resource related information and metadata to be stored and indexed.

Chapter 7

CONCLUSION

7.1 Dissertation Summary

The proposed mathematical algorithms and models enable us to build a Dynamic Light-Weight Knowledge Search (DLKS) platform that makes use of the ever-growing-and-changing web resources.

The backbone of DLKS framework is the domain ontology. The re-ranking algorithm filters and re-sorts the heterogeneous loosely-structured web data into domain relevant knowledge resource. The ontology learning models enable the domain ontologies to grow and self-manage. The framework is quick to build and is well scaled, because it's light-weighted.

The framework is not designed to be an inference engine or a problem solver, so

it is not a knowledge base per se, but rather a knowledge hub that conducts and reroutes traffic of domain knowledge streams coming from the web and consequently, facilitates knowledge learning and knowledge acquisition.

Our algorithms are generic enough to be applied to most knowledge domains.

We based our approaches on two "trusts:" trust to the validity of the seed ontology, and trust to the accuracy of web data crawled by some authoritative general-purpose search engines. The algorithms took the trail that may not have been trodden often by other researchers, which presented the possibility to build domain-dependent ontologies without the influence of an underlining linguistic analysis.

7.2 Future Directions

Although we have seen good results with our proposed algorithms, the experiment was nevertheless preliminary. There is plenty of room for improvement of our algorithms, such as conducting full-life-cycle experiments from new term mining to allocating it on the ontology graph, and obtaining legitimate verification. We also welcome more implementations of and experiments with the algorithms to further the research efforts.

The framework needs lots of work in order for it to complete and improve. It is a theoretical framework. The actual implementation, which will allow for realization

of its worth, has yet to be done.

Appendix A

OBO Basics

The Open Biomedical Ontologies (OBO) flat file format is an ontology representation language. The concepts it models represent a subset of the concepts in the OWL description logic language, with several extensions for meta-data modeling and the modeling of concepts that are not supported in DL languages. The format, comparing to OWL, provides better human readability, ease of parsing, extensibility, and minimal redundancy. Therefore, OBO will be used as our domain ontologies format.

The key terminologies used in OBO are as follow:

Classes/Terms - Classes (often called terms) are the concepts described by the domain ontology.

Relations/Relationship types - Relations/relationship types describe the re-

lationships between terms. *is_a* (subclass) and *part_of* (instance) are both types of relations.

Links/Relationships - A relationship in OBO syntax consists of a child term, a relationship type, and a parent term. Relationships are directional: the child term has a relationship of the given type to the parent term. Because a relationship indicates a directional connection between two terms, they are also called "links."

Bibliography

Agirre, Eneko et al. (2000). "Enriching very large ontologies using the WWW". In: *arXiv preprint cs/0010026*.

Allen, Robert B and Yejun Wu (2002). "Generality of texts". In: *International Conference on Asian Digital Libraries*. Springer, pp. 111–116.

Baader, Franz (2003). *The description logic handbook: theory, implementation, and applications*. Cambridge university press.

Berland, Matthew and Eugene Charniak (1999). "Finding Parts in Very Large Corpora". In: *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. College Park, Maryland, USA: Association for Computational Linguistics, pp. 57–64. DOI: 10.3115/1034678.1034697. URL: http://www.aclweb.org/anthology/P99-1008.

Brown, Peter F et al. (1992). "Class-based n-gram models of natural language". In: *Computational linguistics* 18.4, pp. 467–479.

Burges, Christopher JC (2007). "Ranking as function approximation". In: *Algorithms for Approximation*. Springer, pp. 3–18.

Cimiano, Philipp et al. (2005). "Learning taxonomic relations from heterogeneous sources of evidence". In: *Ontology Learning from Text: Methods, evaluation and applications* 123, pp. 59–73.

Coakes, Elayne (2003). *Knowledge Management: Current issues and challenges*. IGI Global.

Côté, Richard G et al. (2006). "The Ontology Lookup Service, a lightweight cross-platform tool for controlled vocabulary queries". In: *BMC bioinformatics* 7.1, p. 97.

D'Orazio, Vito et al. (2014). "Separating the wheat from the chaff: Applications of automated document classification using support vector machines". In: *Political analysis* 22.2, pp. 224–242.

Feng, Guozhong et al. (2015). "Feature subset selection using naive Bayes for text classification". In: *Pattern Recognition Letters* 65, pp. 109–115.

Finkelstein, Lev et al. (2001). "Placing search in context: The concept revisited". In: *Proceedings of the 10th international conference on World Wide Web*. ACM, pp. 406–414.

Geetha, TV (2017). "Unsupervised Domain Ontology Learning from Text". In: *Mining Intelligence and Knowledge Exploration: 4th International Conference, MIKE 2016, Mexico City, Mexico, November 13-19, 2016, Revised Selected Papers*. Vol. 10089. Springer, p. 132.

Golbreich, Christine et al. (2007). "OBO and OWL: Leveraging semantic web technologies for the life sciences". In: *The Semantic Web*. Springer, pp. 169–182.

Gruber, Thomas R (1993). "A translation approach to portable ontology specifications". In: *Knowledge acquisition* 5.2, pp. 199–220.

Hahn, Udo and Kornél G Markó (2002). "An integrated, dual learner for grammars and ontologies". In: *Data & Knowledge Engineering* 42.3, pp. 273–291.

Hearst, Marti A (1992). "Automatic acquisition of hyponyms from large text corpora". In: *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pp. 539–545.

Huang, Xiangji and Qinmin Hu (2009). "A bayesian learning approach to promoting diversity in ranking for biomedical information retrieval". In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, pp. 307–314.

Järvelin, Kalervo and Jaana Kekäläinen (2002). "Cumulated gain-based evaluation of IR techniques". In: *ACM Transactions on Information Systems (TOIS)* 20.4, pp. 422–446.

Jiang, Jay J and David W Conrath (1997). "Semantic similarity based on corpus statistics and lexical taxonomy". In: *arXiv preprint cmp-lg/9709008*.

Kang, Changsung et al. (2011). "Learning to re-rank web search results with multiple pairwise features". In: *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, pp. 735–744.

Kozareva, Zornitsa and Eduard Hovy (2010). "A semi-supervised method to learn and construct taxonomies using the web". In: *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pp. 1110–1118.

Kozareva, Zornitsa, Ellen Riloff, and Eduard H Hovy (2008). "Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs." In: *ACL*. Vol. 8, pp. 1048–1056.

Krikon, Eyal, Oren Kurland, and Michael Bendersky (2010). "Utilizing inter-passage and inter-document similarities for reranking search results". In: *ACM Transactions on Information Systems (TOIS)* 29.1, p. 3.

Leacock, Claudia and Martin Chodorow (1998). "Combining local context and WordNet similarity for word sense identification". In: *WordNet: An electronic lexical database* 49.2, pp. 265–283.

Lee, Jihyun et al. (2014). "Effective ranking and search techniques for Web resources considering semantic relationships". In: *Information Processing & Management* 50.1, pp. 132–155.

Lempel, Ronny and Shlomo Moran (2001). "SALSA: the stochastic approach for link-structure analysis". In: *ACM Transactions on Information Systems (TOIS)* 19.2, pp. 131–160.

Lenat, Douglas B (1995). "CYC: A large-scale investment in knowledge infrastructure". In: *Communications of the ACM* 38.11, pp. 33–38.

Li, Wei (2011). "Domain-specific information retrieval using rcommenders". In: *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, pp. 1327–1328.

Lin, Dekang (1998a). "An information-theoretic definition of similarity." In: *ICML*. Vol. 98. Citeseer, pp. 296–304.

Lin, Dekang (1998b). "Automatic retrieval and clustering of similar words". In: *Proceedings of the 17th international conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pp. 768–774.

Liu, Wei et al. (2005). "Semi-automatic ontology extension using spreading activation". In: *Journal of Universal Knowledge Management* 1.1, pp. 50–58.

Liu, Yi et al. (2004). "Affinity rank: a new scheme for efficient web search". In: *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, pp. 338–339.

Marchiori, Massimo (1997). "The quest for correct information on the web: Hyper search engines". In: *Computer Networks and ISDN Systems* 29.8, pp. 1225–1235.

Miller, George A (1995). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41.

Navigli, Roberto, Paola Velardi, and Aldo Gangemi (2003). "Ontology learning and its application to automated terminology translation". In: *IEEE Intelligent systems* 18.1, pp. 22–31.

Novalija, Inna, Dunja Mladenić, and Luka Bradeško (2011). "OntoPlus: Text-driven ontology extension using ontology content, structure and co-occurrence information". In: *Knowledge-Based Systems* 24.8, pp. 1261–1276.

Pantel, Patrick and Deepak Ravichandran (2004). "Automatically Labeling Semantic Classes." In: *HLT-NAACL*. Vol. 3, pp. 3–2.

Pereira, Fernando, Naftali Tishby, and Lillian Lee (1993). "Distributional clustering of English words". In: *Proceedings of the 31st annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pp. 183–190.

Resnik, Philip et al. (1999). "Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language". In: *J. Artif. Intell. Res.(JAIR)* 11, pp. 95–130.

Sabrina, T, A Rosni, and T Enyakong (2001). "Extending ontology tree using NLP technique". In: *Proceedings of National Conference on Research & Development in Computer Science REDECS*. Vol. 2001.

Studer, Rudi, V Richard Benjamins, and Dieter Fensel (1998). "Knowledge engineering: principles and methods". In: *Data & knowledge engineering* 25.1, pp. 161–197.

Tam, Vincent WL and John Shepherd (2010). "Webpage relationships for information retrieval within a structured domain". In: *Proceedings of the 21st ACM conference on Hypertext and hypermedia*. ACM, pp. 307–308.

Tang, Bo et al. (2016). "A Bayesian classification approach using class-specific features for text categorization". In: *IEEE Transactions on Knowledge and Data Engineering* 28.6, pp. 1602–1606.

Tang, Duyu, Bing Qin, and Ting Liu (2015). "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification." In: *EMNLP*, pp. 1422–1432.

Vossen, Piek (2001). "Extending, trimming and fusing WordNet for technical documents". In: The Association for Computational Linguistics.

Wächter, Thomas and Michael Schroeder (2010). "Semi-automated ontology generation within OBO-Edit". In: *Bioinformatics* 26.12, pp. i88–i96.

Wu, Zhibiao and Martha Palmer (1994). "Verbs semantics and lexical selection". In: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pp. 133–138.

Yan, Xin, Xue Li, and Dawei Song (2006). "Document generality: its computation for ranking". In: *Proceedings of the 17th Australasian Database Conference-Volume 49*. Australian Computer Society, Inc., pp. 109–118.

Yan, Xin et al. (2011). "Toward a semantic granularity model for domain-specific information retrieval". In: *ACM Transactions on Information Systems (TOIS)* 29.3, p. 15.

Yang, Hui and Jamie Callan (2008). "Learning the distance metric in a personal ontology". In: *Proceedings of the 2nd international workshop on Ontologies and information systems for the semantic web*. ACM, pp. 17–24.

Zhai, Cheng Xiang, William W Cohen, and John Lafferty (2003). "Beyond independent relevance: methods and evaluation metrics for subtopic retrieval". In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, pp. 10–17.

Zhang, Yudong et al. (2014). "Binary PSO with mutation operator for feature selection using decision tree applied to spam detection". In: *Knowledge-Based Systems* 64, pp. 22–31.

Zhao, Grace and Xiaowen Zhang (2017). "A Domain-Specific Web Document Reranking Algorithm". In: *Advanced Applied Informatics (IIAI-AAI), 2017 6th IIAI International Congress on*. IEEE, pp. 385–390.

Zhao, Grace and Xiaowen Zhang (2018). "Domain-Specific Ontology Concept Extraction and Hierarchy Extension". In: *The 4th ACM SIGIR International Conference on the Theory of Information Retrieval*. ACM (Submitted).

Zhao, Grace, Xiaowen Zhang, and Abdullah Uz Tansel (2014). "Open km-Learning: Using Semantic Web Technologies to Facilitate E-Learning". In: *Proceedings of the International Conference on Semantic Web and Web Services (SWWS)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), pp. 1–5.