A Comparison of Taxonomy Generation Techniques Using Bibliometric Methods: Applied to Research Strategy Formulation

by

Steven L. Camiña

S.B., E.E.C.S. M.I.T., 2009

Submitted to the Department of Electrical Engineering and Computer Science

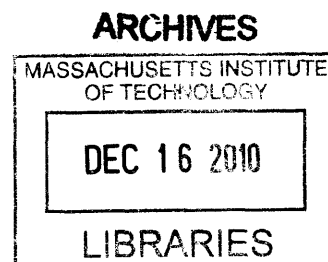in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

July 2010
[September 2010]

Copyright 2010 Steven L. Camiña. All rights reserved.

Author_____
Department of Electrical Engineering and Computer Science
July 23, 2010

Certified by_____
Stuart Madnick
John Norris Maguire Professor of Information Technologies and
Professor of Engineering Systems, Massachusetts Institute of Technology
Thesis Co-Supervisor

Certified by_____
Wei Lee Woon
Assistant Professor, Masdar Institute of Science and Technology
Thesis Co-Supervisor

Accepted by_____
Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

1

A Comparison of Taxonomy Generation Techniques
Using Bibliometric Methods:
Applied To Research Strategy Formulation
by
Steven L. Camiña

Submitted to the
Department of Electrical Engineering and Computer Science

July 23, 2010

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

This paper investigates the modeling of research landscapes through the automatic
generation of hierarchical structures (taxonomies) comprised of terms related to a given
research field. Several different taxonomy generation algorithms are discussed and
analyzed within this paper, each based on the analysis of a data set of bibliometric
information obtained from a credible online publication database. Taxonomy generation
algorithms considered include the Dijsktra-Jarnik-Prim's (DJP) algorithm, Kruskal's
algorithm, Edmond's algorithm, Heymann algorithm, and the Genetic algorithm.
Evaluative experiments are run that attempt to determine which taxonomy generation
algorithm would most likely output a taxonomy that is a valid representation of the
underlying research landscape.

Thesis Co-Supervisor: Stuart Madnick
Title: John Norris Maguire Professor of Information Technologies and Professor of
Engineering Systems, Massachusetts Institute of Technology

Thesis Co-Supervisor: Wei Lee Woon
Title: Assistant Professor, Masdar Institute of Science and Technology

Table of Contents

List of Figures

List of Tables

CHAPTER 1: Introduction

1.1 Motivations

1.1.1 Experts and the Decision Making Process

Decision making is a cognitive process resulting in the selection of a course of action among several alternatives, usually relying on the opinions of qualified authorities and led by subject-matter experts whose experience and internalized knowledge allow for effective decisions to be made. Experts usually work within a given research field and are deeply immersed in their subject of expertise. This allows them to give credible advice to researchers. However, in the end, one expert cannot possibly know all the information that exists relating to their field at all times. An expert may not have complete information about a field of technology or research, since the landscape is constantly changing. Everyday, new technologies are invented, outdated research methodologies scrapped, and research strategies altered and improved. It is difficult for an expert to constantly keep track of all of these developments.

Experts are also human, hence decisions made by them will be partially based on their own personal perspectives and unique experiences in the field. As a result, expert advice is still somewhat subjective in nature.

Expert input is extremely valuable to the decision-making process. With this in mind, one issue that motivated the work in this thesis was aiding the decision-making process by helping experts acquire a more complete understanding of their area of expertise.

1.1.2 Research Landscapes

Every research field is composed of a set of interrelated concepts / ideas. For example, within the research field of "renewable energy", there are several interrelated concepts such as "solar power", "hydroelectric power" and "electricity". Going a level deeper, within "solar power", there are also several interrelated concepts such as "photovoltaics" and "thermovoltaic". We collectively refer to the set of interrelated concepts within a given research field as its *research landscape*.

In technology-intensive sectors, decision-makers and researchers are always looking for new, better ways to understand their field. A clear understanding of a research landscape will help give their research direction, purpose, and can also help justify its need to investors who, at the end of the day, provide the monetary incentive for continuing research.

A research landscape is not static, but rather changes constantly as new technologies and concepts emerge, almost on a daily basis. Another issue that motivated the work in this thesis was to accurately generate a robust visualization of a research landscape that provides useful information to those that view it.

1.1.3 Analysis of Publication Databases

Text data mining refers to the process of gathering information from text through searching for patterns / trends. Typically, the text to be analyzed is first parsed, structured, and cleaned up, then the output is evaluated using various statistical techniques. Text data mining is frequently applied to publication databases. A publication database refers to an organized set of data composed of documents, articles, and entries gathered from journals, magazines, conference proceedings, blogs, and other publicly released collections. Several publication databases exist, many of which are readily available online. Ever since the Internet became mainstream, the volume of useful information available online has increased exponentially. Online publication databases have been developed to help manage the vast amounts of information, yet even with these it is still hard to decipher which bits of information are worth examining and which are just a waste of time.

There are several academic online publication databases that specifically review technologically-related journals, such as Compendex and Inspec (collective called Engineering Village), Scirus, Scopus and Web of Science. These databases contain an extraordinary amount of information for any individual to read, comprehend and process.

Another issue that motivated the work in this thesis was methodologically extracting all the information in these publication databases without the need of manual inspection and presenting the information to end-users in a simple, easily-understandable medium.

1.2 Technology Forecasting Using Data Mining and Semantics

With all these motivations in mind, our team at MIT, in cooperation with a team in the Masdar Institute of Science and Technology (MIST), have been developing an automated method of helping technologically oriented decision makers make more informed decisions. The idea was to solve the three problems mentioned in the previous section: aiding experts in giving credible advice, visualizing research landscapes, and sifting through information in publication databases, all with one tool.

MIT and Masdar have been collaborating these past two years on a project that aims to mine science and technology databases for patterns and trends which can facilitate the formation of research strategies [Woon et al. 2009(1)]. Examples of the types of information sources are academic journals, patents, blogs and news articles. The proposed outputs of the project were:

1. A detailed case study of the renewable energy domain, including tentative forecasts of future growth potential and the identification of influential researchers or research groups

2. An improved understanding of the underlying research landscape, represented in a suitable form, like a taxonomy

3. Scholarly publications in respected and peer-reviewed journals and conferences relating to the research

4. Software tools to automated the developed techniques.

The high-level aim of the project is to create improved methods for conducting *technology mining* using *bibliometric* techniques. Technology mining refers to the process of gathering information from publication databases of technological literature. Bibliometrics refers to the statistical analysis of a document without the actual extraction of each document's fulltext.

The basic framework of the entire project is shown in Figure 1.
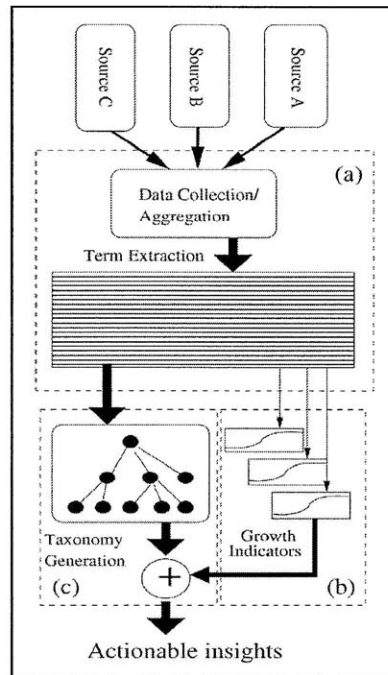
Figure 1: Technology Forecasting Using Data Mining and Semantics Project Framework

Notice that the figure is composed of several distinct blocks. Each block represents a separate phase in the system. Block (a) represents data collection / aggregation and term extraction. In this phase, bibliometric information is extracted from a publication database and a list of key terms is collected on which the technology forecasting efforts will be focused. Block (b) represents the identification of early growth technologies. There are two steps to this phase. The first is to find a suitable measure for the 'prevalence' of a given technology as a function of time, and the second is to locate technologies that, based on this measure, appear to be

technologies in the "early growth" phase of their development. Finally, Block (c) represents the phase where terms are visualized using a predictive *taxonomy*, described later.

1.3 Project Objectives

The work presented here is a subset of the work described in the previous section. Specifically, the work here focuses on the second goal of the broad project mentioned previously: an improved understanding of the underlying research landscape, represented in a suitable form, like a *taxonomy*.

The underlying assumption to our work is that a research field can be divided into distinct, yet interrelated terms, which are words / word phrases that embody a specific concept. These terms make up the research landscape, as described earlier. We believe that we can find these terms and determine their relation to each other by parsing the information contained in an online publication database. In the succeeding chapters, we describe a process for automatically gathering key terms related to a technological field from a publication database and organizing these terms into a structure called a *taxonomy*, which is a hierarchical organization of terms relevant to a particular domain of research, where the growth indicators of terms lower down in the taxonomy contribute to the overall growth potential of higher-up "concepts" or categories. The ordering of the terms in the taxonomy should reflect the inter-relationships between the terms in the context of the research field being examined.

A *taxonomy* is an acyclic graph where each node has exactly one incoming edge but can have multiple outgoing edges. For the purposes of research landscape taxonomy generation, each node in the taxonomy is a term / concept in the research field. An example of a taxonomy generated from a hypothetical research landscape of "renewable energy" is shown in Figure 2.
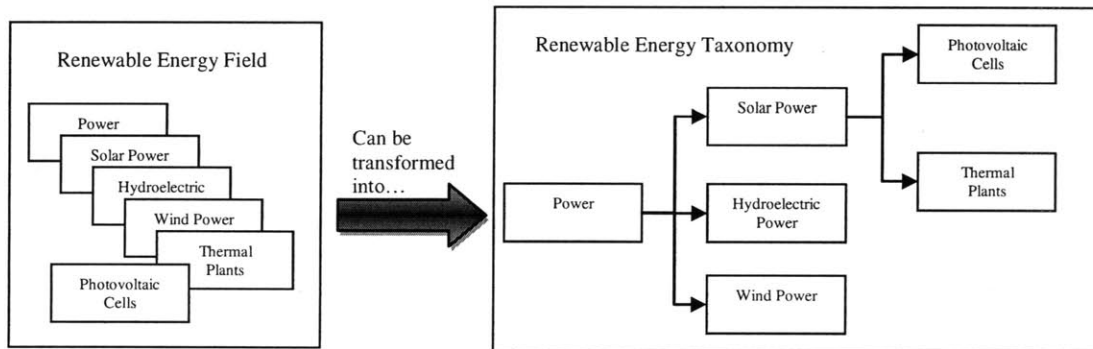
Figure 2: Generating a Taxonomy from a Technological Field Landscape

The box on the right of Figure 2 shows a taxonomy based on the technological field shown in the box on the left. It can be seen that there is only one unique path between each

technological concept / term. We believe that a taxonomy is a very effective representation for visualizing research landscapes because:

1. The unique paths that can be traced between pairs of terms show clear conceptual links amongst terms.

2. Automatically generated taxonomies reflect the information contained in thousands of published academic papers, reflecting the opinions of many well-respected authors who have published papers in the field.

In this thesis, we evaluated methods based on mathematically-grounded algorithms that utilize the vast amount of information found in scientific and technological academic publication databases to generate a sensible taxonomy representing a research field. Motivated by the issues stated in Chapter 1.1, the overall goals of this thesis are:

1. To develop automated, publication database-independent methods.

2. To compare several taxonomy generation algorithms and evaluate the usefulness of each.

3. To generate ways of visually representing taxonomies in a manner that is easily understandable for viewers.

4. To run a case study on "renewable energy".

1.4 Overview

The rest is structured as follows:

Chapter 2 will review the academic literature relating to taxonomy generation.

Chapter 3 will go in depth regarding the steps involved in the taxonomy generation process

Chapter 4 will discuss the methodology for evaluating taxonomy generation algorithms .

Chapter 5 will present the results of running the analyses described in Chapter 4.

Chapter 6 will wrap up the analysis and discuss where future work can be done.

CHAPTER 2: Literature Review

2.1 Technology Forecasting

Technology forecasting is of particular importance to the research presented in this thesis because our work in research landscape visualization facilitates technology forecasting. Many academics in the field have also investigated problems relating to tech forecasting and have tried to address them. In proof, there is already a significant body of related research on the subject. This rest of this subsection first presents related literature to technology forecasting, then discusses how our work complements the existing body of research.

[Porter 1991] discussed general issues related to forecasting and management, and introduced some basic tools for quantitative technological trend extrapolation. The book elaborated on the planning, operation, analysis and control of complex technological systems and new technology. The book covers the basics for long term planning, new product development and production, and shows the factors that must come together for new technologies to be developed and new complex products to be produced. Using exhibits, and case studies, [Porter 1991] discusses the methods for dealing with significant issues in managing technological development.

Another book from the same author, [Porter 2005] focused specifically on the process of technology mining, which is the process of extracting usable information from patents, business information and research publications for the purpose of aiding the management of technology (MOT) process which has thusfar largely been intuition-driven. Technological sources of information are treated as the data that will eventually be "mined" in order to aid the MOT process and generate conclusions about the field of interest. The tech mining analysis described in [Porter 2005] looked at when was the research done, where was it patented, who were the major organizations involved, what were the technological areas of focus, who were the leaders of the companies involved, and what is the current state of the tech industry. It then created matrices showing co-occurrences between these fields in the data, then looked at the change in the data over time to finally generate some conclusions about the technological field.

[Martino 1993] is one of the most widely cited texts in technology forecasting literature. It defined a technological forecast as a prediction of future characteristics of machines, procedures or techniques. It then presented technological forecasting as an aid to decision-making by presenting a comprehensive overview of forecasting methods, using numerous real-world examples and illustrations.

The works presented in this subsection show that technology forecasting as a body of research is indeed promising and a lot of utility can be derived from any tool or methodology that could move this body of research forward. However, in none of the works reviewed and presented here could we find any attempt to use technology mining methods provide a clear, concrete set of actions for decision-makers and researchers. We believe that this is a critical gap worth investigating and the research in this thesis seeks to address this issue. We know this is a

challenging task, but we believe that one such way to help accomplish this is the visualization of a research landscape, which we do in the form of a taxonomy. The other research featured in this section either simply define tech mining as a research field or present tech mining techniques for managing technological development. None of them actually present any tech mining methods whose outputs can be of immediate actionable impact. For researchers and decision-makers that view the taxonomies we generate, it is immediately clear what concepts they should be focusing on within a given technological field, which we believe could give them insights on what actions to take especially in the early stages of decision-making, where the researchers usually have a lesser understanding of the technological field as a whole.

2.2 Taxonomy Generation

There have been several previous studies on taxonomy creation for various purposes. [Blaschke 2002] proposed a method that automatically generated classifications of gene-product functions using bibliometric information, which is then compared to the well accepted GO ontology. [Krishnapuram 2003] talked about the issues and possibilities concerning automated taxonomy generation. In particular, the paper reviewed several taxonomy generation approaches and provided insight into the issues involved. [Sanchez 2004] presented a methodology to extract information from the internet and build a taxonomy of terms and web resources for a given domain. [Schwarzkopf et al. 2007] proposed an approach for using data from a social tagging application, like del.icio.us as a basis for user adaptation, then mining taxonomies of tags from tag spaces. The mined taxonomy can then be used to determine how to adapt a system to a user given the user's personal tag space. [Chuang et al. 2002] discussed automatic query clustering, specifically, organizing query terms into a hierarchical structure, producing a query taxonomy.

We believe that the research concept proposed in this thesis is novel because unlike any other study, we use taxonomy generation for the specific purpose of generating output that helps facilitate decision-makers and researchers, which has not been done in any of the works of literature mentioned in the last paragraph. In this thesis, we devise methods for automatically generating solid, reliable taxonomies.

2.3 Bibliometric Analysis

There has been lots of work dealing with bibliometric analysis, presented in this subchapter.

[Kostoff 2000] showcased database tomography, a bibliometric database analysis system that extracts multi-word phrase frequencies and proximities in order to augment analysis from experts in a research field. [Kostoff 2001] then followed up by describing an approach for identifying pathways through which research can impact other research, technology development / applications, and technical / infrastructure characteristics of the user population using citation analysis.

14

[Okubo 1997] presented the essential elements of bibliometrics and its application to the analysis of research systems. It started by describing the advent of bibliometrics, continuing with the presentation of the main bibliometric databases that existed back when the paper was written, the bibliometric indicators, and the ways to apply these indicators.

[Small 2006] looked at using co-citation clusters to track the growth and emergence of research areas in science. It defined a research area as a set of documents that define a research topic and an associated group of researchers who share an interest in the topic. Then, it talked about the methodologies of co-citation clustering, mapping, and string formation, and defined a measure of cluster relativity based on the change in average age of highly cited papers.

In addition to the work mentioned above, there have been other works such as [van Raan 1996], [Daim 2006], [Verbeek 2002], and [Narin 1996] relating to bibliometric research. Moreover, there is some research that talks about the need for standards in bibliometric research [Glanzel 1996], and methods to mine text using keyword distributions [Feldman 1998]. Within this large body of literature however, none of the works deals directly with using bibliometric analysis in order to output something that could be used in technology forecasting and decision-making facilitation, much like how we use bibliometric analysis to output a taxonomy that experts can directly gather information from.

The work in [Ziegler 2009] uses bibliometric analysis intensively and served as the springboard for the work in this thesis. Here, automated methods for bibliometric analysis using information from online publication search engines were developed. The work in [Ziegler 2009] took as input a word or phrase representing a research field, which is called a *seed term*, and attempted to:

1. Discover related technologies / keywords to the *seed term*

2. Calculate a numerical value for the growth rate of a certain technology within the research field encapsulated by the *seed term*, in hopes of flagging fast-growing technologies, which could then be relayed to experts

3. Determine the relationships among technologies within the research field encapsulated by the *seed term* by grouping them into "concept" clusters

4. Identify new, upcoming technologies within the research field encapsulated by the *seed term*

The first stage of [Ziegler 2009]'s analysis is keyword extraction. Given an initial *seed term*, online publication databases such as Compendex and Inspec[1] and Scirus[2] were scanned and some/all of the keywords that come up as "related terms" were extracted. Each online publication database presents its information in different ways, and hence unique "wrapper code" was developed for each website. The choice of databases to query is based on two important criteria:

---

[1] Available via www.engineeringvillage.com
[2] Available via www.scirus.com

first, the database must index a large number of articles related to the technological field of interest, and second, each database must present its results in a consistently formatted way in order to allow keywords to be scraped in an automated fashion.

[Ziegler 2009] also attempted to refine the keywords extracted from the Scirus database by using back-pointing and eigenvector centrality. The concepts of back-pointing and eigenvector centrality help in ensuring that the "related terms" produced after a search are actually relevant. Relevance fitting by back-pointing works by imposing a restriction on the terms extracted: they must all "point back" to the original seed term. In other words, after a number of nested searches of listed related keywords, the original seed term must be reached. If this condition is not satisfied, then it is assumed that the term is not really related to the seed term. There are a few variants to the back-pointing algorithm that will not be discussed within this paper. Relevance fitting using eigenvector centrality simply means viewing all the terms in Scirus as a densely connected network, where a link exists between terms that are related to each other. Given this, the eigenvector centrality, which is a measure of importance / connectivity, can be calculated for each term. The terms that have a lower centrality value are then disregarded.

[Ziegler 2009] then extracted *hit counts* from each online database, which represents the number of articles related to a seed term for each year. These *hit counts* are then extracted and saved, and later used to calculate the *growth rates* for each of the terms. The growth rates are used to rank the list of terms. Terms with high growth rates and a relatively small number of current hit counts are considered as potentially "high growth" terms that could well be part of mainstream research in the future.

Finally, [Ziegler 2009] used Latent Semantic Analysis (LSA) in order to cluster terms into manageable "concepts". Often, the output of a search produced keywords that were very closely related, such that they could regarded as synonyms. Based on the co-occurrence of terms in documents, the LSA algorithm produces a set of "concepts", each of which is a weighted combination of every term in the field. LSA is based on a well-known and commonly-used technique in linear algebra called Principal Component Analysis. An additional use of LSA is to aid in term cleaning, where terms that do not have a strong weighting within any of the concepts generated by LSA are discarded.

The work in this thesis is largely a continuation of the work in [Ziegler 2009]. In [Ziegler 2009], *bibliometric analysis* and *technology mining* techniques were used to generate and filter terms. The work in this thesis takes things a step further. While [Ziegler 2009] stopped at term collection and concept creation, the work presented here takes the terms generated using the technology presented in [Ziegler 2009] and organizes them into a taxonomy, which we believe could be used to aid decision-makers and researchers. The work developed and presented in this thesis both developed taxonomy generation methods and evaluated each taxonomy generation algorithm's usefulness.

CHAPTER 3: Taxonomy Generation Process

The previous two chapters described the goals and aims of this research project. In particular, the chapters discussed the overall goal of the MIT / MIST research group and this thesis which is the automated creation of accurate, reliable taxonomies. However, this goal is still quite broad and hence the specific focus of this thesis is evaluating algorithms used for taxonomy generation. This chapter will explain the entire taxonomy generation process we have developed and present each of the algorithms in detail.

Before the actual taxonomy generation algorithm can be run, each of the taxonomy generation algorithms needs to be given inputs of a specific form, which in turn are based on bibliometric information contained in a publication database. For the purposes of this thesis, we collect the bibliometric information from an online publication database, but in theory the same information can be collected from one that is not online.

Each taxonomy generated is centered around a particular technological concept, summarized in a term or phrase or group of phrases called seed terms. The initial choice of initial seed terms is necessarily made by the user. As a case study within this thesis, we used seed terms related to "renewable energy", but these are not the only terms that can be used.

Using bibliometric information collected from online publication databases as a basis, the taxonomy generation algorithms we developed determine which terms to logically link together in the final taxonomy.

The choice of a suitable publications database to gather information from is critical, as each different publication database contains a collection of articles from several different sources, which may or may not be within the scope of the research area we are investigating. For instance, a database like CHEMnetbase[3] that contains articles from journals relating only to "chemistry"[4] is not going to be a very good resource when searching for articles related to "renewable energy".

3.1 Chapter Overview

The general procedure for taxonomy generation will be discussed in the succeeding sections. As mentioned previously, the taxonomy generation algorithms we've developed take a certain type of input which is derived from publication databases. As such, this chapter will first discuss how this input is generated before tackling the specifics of each taxonomy generation algorithm.

The steps necessary before the taxonomy generation algorithms can be run are:

---

[3] Available via www.chemnetbase.com
[4] "Chemistry" in this context refers to the science dealing with matter and its changes

1. Bibliometric information is extracted from an publication database, which in the case of this thesis is found online, and stored locally for quick processing.

2. From the bibliometric data, a set of terms are chosen amongst the article keywords that are to be included in the taxonomy.

3. A similarity measure is used to compare chosen terms to be included in the taxonomy by quantifying each pair of terms' relationship strength, collectively represented by a distance matrix.

Once the distance matrix has been generated, taxonomy generation algorithms can be run. In particular the taxonomy generation algorithms that we've developed and will be discussed in the succeeding sections are:

1. Dijsktra-Jarnik-Prim's (DJP) Algorithm

2. Kruskal's Algorithm

3. Edmond's Algorithm

4. Heymann Algorithm

5. The Genetic Algorithm

Finally, the outputted taxonomy must be presented in an aesthetically pleasing manner. As such, the way in which we visualize taxonomies is discussed at the end of the chapter.

3.2 Extracting Bibliometric Information

The first key step in taxonomy generation is the extraction of bibliometric information from publication databases via their respective search interfaces. Bibliometric information refers to the data pertaining to the low level statistical properties of an article, as opposed to the actual contents; note, that the extraction of this information may still require that the text of entire documents be parsed – however, this will only be to extract these statistics and not, for example, to conduct higher level analyses such as natural language processing. Specifically, the bibliometric information we analyzed were the 'title', 'abstract', and 'keywords' of an article.

To gather information from an online publication search engine, the seed term(s) of choice were first entered into the database's search interface (this can either be done manually, or via some automated procedure or API). Information regarding matching documents was then retrieved, allowing the extraction of the relevant bibliometric information.

While it is unlikely that a single publication database would be able to cover all relevant academic journals, we have found several that we believe cover topics that most closely relate to the research landscape that we are exploring in this project, "renewable energy". While we cannot

be certain if the databases we found are necessarily the best databases for our specific purpose, we know that these databases are very highly regarded, cover a wide scope of topics relating to technology, and are readily available without additional cost within MIT / MIST.

3.2.1 Engineering Village

In MIT, the database of choice is Engineering Village[5]. Engineering Village is a combination of three online databases: Compendex, Inspec and NTIS. Compedex and Inspec are both significantly larger in scope compared to NTIS (National Technical Information Service). The latter is a database of government reports and information covering several product categories ranging from administration/management to earth sciences. Because of NTIS's limited scope compared to Compendex and Inspec, we focused our data gathering efforts on Compendex and Inspec. Compendex and Inspec cover publications from 1884 up to the present and are available free of charge to members of the MIT community, allowing our research group to query the online publication database as often as we wanted without any overhead.

Compendex is a comprehensive bibliographic database of scientific and technical engineering research, covering all engineering disciplines. It includes millions of bibliographic citations and abstracts from thousands of engineering journals and conference proceedings. Compendex covers well over 120 years of core engineering literature. Specifically, Compendex includes over 5 million summaries of journal articles and conference proceedings and 220,000 new additions every year. Over 5,000 engineering journals and conferences are indexed and the database is updated weekly. Coverage of Compendex includes: Mechanical Engineering, Civil Engineering, Electrical Engineering and Electronics, Chemical Engineering and Aeronautical Engineering. Compendex is produced by Elsevier Engineering Information Inc.

Inspec includes bibliographic citations and indexed abstracts from publications in the fields of physics, electrical and electronic engineering, communications, computer science, control engineering, information technology, manufacturing and mechanical engineering, operations research, material science, oceanography, engineering mathematics, nuclear engineering, environmental science, geophysics, nanotechnology, biomedical technology and biophysics. Inspec contains over eight million bibliographic records taken from 3,000 scientific and technical journals and 2,000 conference proceedings. Over 400,000 new records are added to the database annually. Online coverage is from 1969 to the present, and records are updated weekly. Inspec is produced by the Institution of Engineering and Technology (IET).

Compendex and Inspec are similar in a few ways. First, although they mostly cover a different set of topics, they do have around a 20% overlap of journals between them. Also, since they are both contained in the Engineering Village website, they both display "controlled terms" and "uncontrolled terms" for each article. "Controlled terms" come from the controlled vocabulary found in the EI Thesaurus, which is used to index records in Compendex. EI refers to

---

[5] Available via www.engineeringvillage.com

Engineering Information, which is a business unit of Elsevier[6], which is one of the leaders in providing online information, knowledge and support to engineering researchers. The 4th edition of the EI Thesaurus contains 18,000 terms and EI's controlled vocabulary is a list of subject terms used to describe the content of a document in the most specific and consistent way possible. "Uncontrolled terms" are author imposed keywords for the article. The number of "controlled terms" and "uncontrolled terms" for each article ranges, but typically each article has around 5 controlled terms and anywhere between 5 to 20 uncontrolled terms.

Collecting Terms from Engineering Village

Given an initial seed term, Engineering Village is queried via its online interface and the bibliometric information of all the articles produced in the search results is stored in a locally stored database file.

The initial page in Engineering Village is shown in Figure 3. To query the database, the search term is typed into the designated text box enclosed in double quotes to ensure that the seed term is treated as a single phrase rather than a set of disjoint words. For example, we would type in ["renewable energy"] as opposed to [renewable energy]. The correct checkbox is also selected to indicate which database among Compendex, Inspec will be used.
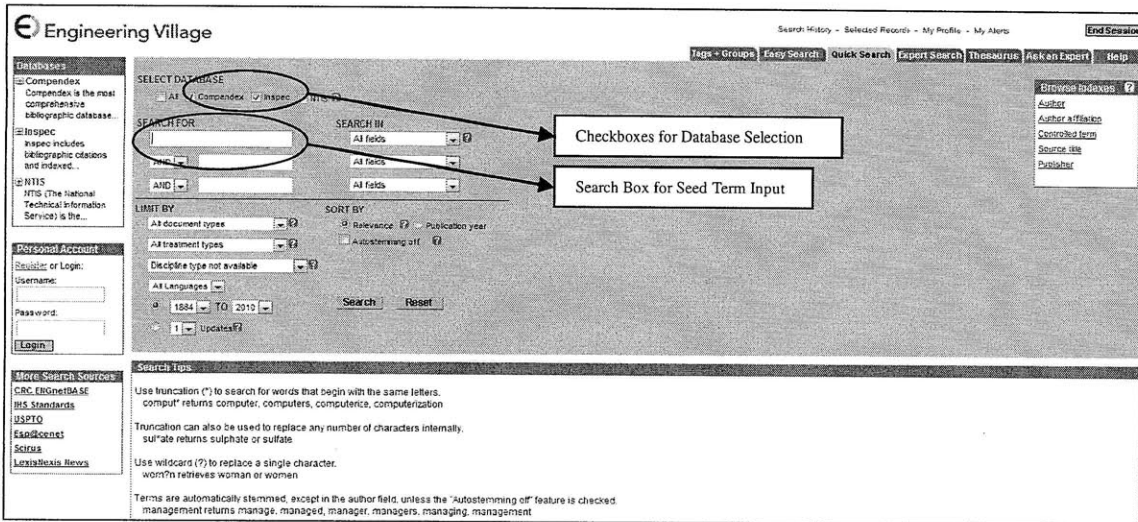
Figure 3: Home page of Engineering Village

After the search is run, the results appear in a new browser page similar to the one shown in Figure 4.

Figure 4: Typical Search Results page for Engineering Village with the key sections highlighted. Notice that near the top of the page, the number of total records is displayed. Also, the list of the first 25 relevant results appear throughout the page with a link to the "Next" 25 at the bottom right of the page. Each relevant article has a link that redirects the user to another page that shows its abstract, which contains the article's title, author, abstract, keywords and citations.

Clicking on the "Abstract" button for each article links to a page similar to the one shown in Figure 5.
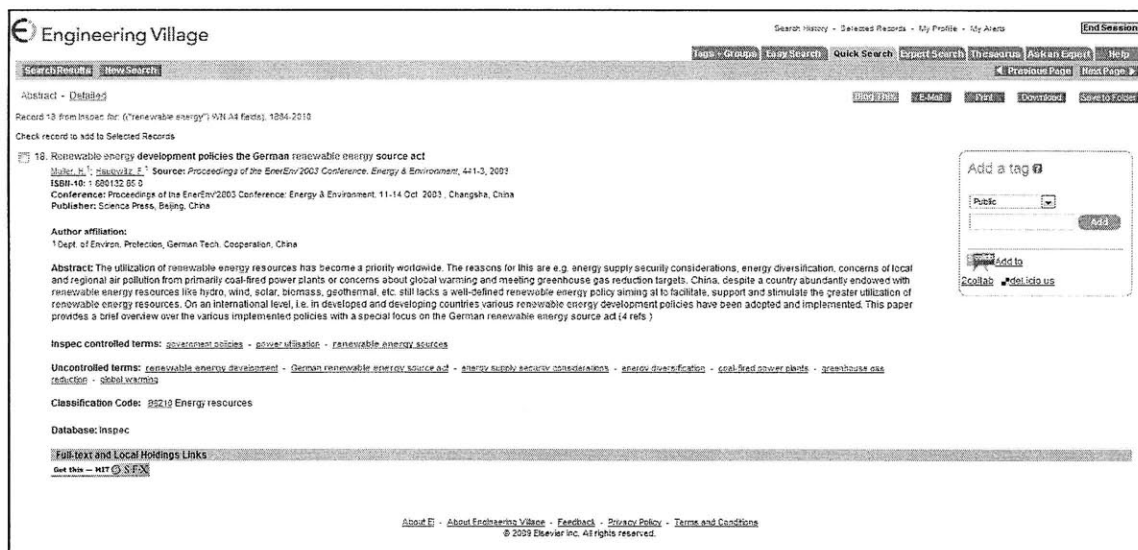
Figure 5: Detailed Abstract Page for Each Article.
This page contains more detailed information about a specific article, including its Title, Author(s), Publication Source, Publication Date, Abstract, Controlled Terms, Uncontrolled Terms, and Classification Code

To extract all the relevant information, the database is first queried with the seed term, which is equivalent to typing in this lengthy URL:

http://www.engineeringvillage.com/controller/servlet/Controller?CID=quickSearchCitationFormat&database= **<database number: 1 for Compdendex, 2 for Inspec>**
&searchWord1= **<seed term/phrase separated by "+" signs instead of spaces">**
section1=NO-LIMIT&boolean1=AND&searchWord2=&section2=NO-
LIMITboolean2=AND&searchWord3=&section3=NO-LIMIT&doctype=NO-
LIMITtreatmentType=NO-LIMIT&disciplinetype=NO-LIMIT&language=NO-
LIMITsort=relevance&yearselect=yearrange
&startYear= **<start year to search for publications>**
&endYear= **<end year to search for publications>**
stringYear=CSY1884CST1884ISY1896IST1896NSY1899NST1899updatesNo=1&search.x=23&search.y=12&search=Search

Searching for patterns, or "regular expressions", within the convoluted source code of the results page, the URLs for the abstracts of each article are extracted. Then, each abstract URL is read individually, and the "title", "abstract", "controlled terms", and "uncontrolled terms" of each article are captured by searching for more regular expression patterns throughout the page.

After all the data from the abstracts within a specific results page are gathered, the URL of the next page containing the next 25 articles is gathered and the process is repeated for the articles / results within that page.

The data extraction process does not make many queries to the actual Engineering Village website. From the website's perspective, the data collection operation functions exactly like regular searches, where the "Next Page" button is clicked on multiple times. This is in

contrast to an earlier approach where data was gathered individually for each search term by continually querying the site and gathering the total number of results or "hit counts" produced by each search [Ziegler 2009]. This approach resulted in the generation of a very large number of requests to the remote website, which meant that we not only ran the risk of having our IP address blocked, we also could only run the taxonomy generation algorithm while connected to the internet and from within the MIT network. The current approach downloads and collects the relevant data from the remote website in one quick sweep, permitting all subsequent computations to be performed offline.

The use of Python's regular expression and URLLib packages allowed for the easy extraction of online data. There was one slight complication to the data gathering process, which was that Engineering Village only provided up to 4,025 articles per query, despite listing much more in its article count estimate at the top of the webpage. This meant that visiting the URL for the page after the 4,025th article would display an error. To get around this, the whole database was collected by conducting several incremental queries where the results of each were limited by altering the "start year" and "end year" parameters to extract only a subset of data each time, and then all the incremental search results were aggregated to produce one massive database. This allowed for the collection of bibliometric information from hundreds of thousands of articles.

All the bibliometric data taken from Compendex / Inspec was then stored in a local SQLite3 database, chosen for its lightweight and easily transferrable properties. The database has the following schema:

- TITLE

- ABSTRACT

- CONTROLLED TERMS

- UNCONTROLLED TERMS

- JOINT TERMS

  - representing the union of the controlled and uncontrolled terms

The core terms that will be used to populate the final taxonomy are gathered from either the CONTROLLED_TERMS, UNCONTROLLED_TERMS, or JOINT_TERMS.

3.2.2 Scopus

In Masdar (MIST), the database of choice is Scopus. Scopus is the largest abstract and citation database of peer-reviewed literature and quality web sources. Updated daily, Scopus offers nearly 18,000 titles from more than 5,000 international publishers, including coverage of 16,500 peer-reviewed journals, 600 trade publications, 350 book series and 3.6 million

23

conference papers. Scopus contains over 40 million records going back as far as 1823. Scopus covers topics in science, technology, medicine, and social science. 80% of all Scopus records have an abstract.

Scopus has a convenient feature where all the bibliometric information contained in the search results can be downloaded with a few simple clicks. As such, the data gathering process from Scopus was much simpler than the process for the Engineering Village databases. The bibliometric information downloaded from Scopus comprised of the following information for each relevant article produced by querying the site:

- AUTHORS
- TITLE
- YEAR
- SOURCE_TITLE
- VOLUME
- ISSUE
- ARTICLE_NUMBER
- PAGE_START
- PAGE_END
- PAGE_COUNT
- CITED_BY
- LINK
- AFFILIATIONS
- AUTHORS_WITH_AFFILIATION

- ABSTRACT
- AUTHOR_KEYWORDS
- INDEX_KEYWORDS
- CORRESPONDENCE_ADDRESS
- EDITORS
- PUBLISHER
- ISSN
- ISBN
- CODEN
- DOI
- LANGUAGE_OF_DOCUMENT
- ABBREV_SOURCE_TITLE
- DOCUMENT_TYPE
- SOURCE

The data gathered was then stored in an SQLite3 database, similar to Engineering Village. The core terms that are then used to populate the final taxonomy are gathered either from the AUTHOR_KEYWORDS or INDEX_KEYWORDS. AUTHOR_KEYWORDS are similar to the "uncontrolled terms" in Compendex / Inspec, while INDEX_KEYWORDS are similar to the "controlled terms".

For this thesis, bibliometric information pertaining to "renewable energy technologies was collected from Scopus and stored in a local SQLite3 database. This local database, which we refer to in all succeeding parts as the backend *data set* compiled for a given seed term is not fed into the taxonomy generation algorithms just yet. An additional transformation needs to be done to the information first. The process of converting the raw database information into a workable form is discussed in the succeeding sections.

3.3 Quantifying Term Similarity

In order to process the data, concepts from graph theory were used. In computer science, graphs are mathematical structures used to model pairwise relations between objects in a given set. A graph contains a collection of 'vertices' or 'nodes' (used interchangeably in this paper) and a collection of 'edges' or 'links' (also used interchangeably in this paper) connecting pairs of nodes. A graph may be undirected, which means there is no distinction between the two nodes associated with each edge, or directed, in which case each edge specifies a path from one node to another. Several of the taxonomy generation algorithms developed and used in this paper are based on existing graph theory algorithms.

The first step in processing the data from the data set is to convert it into a workable graph. This graph is called a term similarity graph, where the nodes of the graph represent individual terms and the edges between the nodes represent the strengths of their relationship with each other. A key intuition behind our approach is that the relationship between terms in the taxonomy can be quantified based on the frequency that these terms occur simultaneously in academic literature. Simply put, we assume that the repeated appearance of a specific keyword pair in several different articles implies a close relation between the terms. Building on this premise, we calculate the 'relationship strength' between each pair of nodes based on a similarity metric that took as primary input the frequency in which pairs of terms co-occurred within the bibliometric information of each article in the database. A co-occurrence between a pair of terms is defined as the co-existence of two terms within a particular article's title, abstract, or keywords.

Based on literature on the subject, we decided to use these forms of similarity: cosine similarity, symmetric and asymmetric normalized google distance (NGD) similarity. Cosine similarity and symmetric NGD produce a metric that is undirected between terms, whereas the asymmetric NGD metric produces a directed term strength metric.

3.3.1 Cosine Similarity

Cosine similarity is a measure of similarity between two vectors based on the cosine of the angle between them. This method is often used to compare documents in text mining, and connectedness within clusters in data mining. Given two vectors A and B, cosine similarity is defined s:

$$cosine\ similarity = \frac{A \cdot B}{\|A\| \|B\|} \qquad \text{(Eq. 1)}$$

Applied to taxonomy generation, this can be rewritten as:

$$cosine\ similarity = \frac{n_{x,y}}{\sqrt{n_x} \cdot \sqrt{n_y}} \qquad \text{(Eq. 2)}$$

where n_x and n_y represent the number of articles that contain terms x and y respectively, and $n_{x,y}$ represents the number of articles that contain both x and y. An article is said to 'contain' a term if the term occurs within its title, abstract, or list of keywords.

Application of the formula in Eq. 2 results in a cosine similarity value of between 0 and 1, where 0 means independent, and 1 means exactly similar. Also, note that the cosine similarity between two terms is symmetric. This means that the similarity of term a to term b is the same as the similarity of term b to term a. Because of this, when applied to taxonomy generation, the cosine similarity of a pair of terms does not give a clear indication regarding which of the terms in the pair will be the child of the other in the final generated taxonomy.

3.3.2 Symmetric Normalized Google Distance Similarity

The symmetric normalized Google[7] distance (NGD) similarity between terms is based on [Cilibrasi & Vitanyi 2007]. In their work, they described Google Distance as a method that uses term co-occurrence frequencies to indicate how close terms are related to each other. The closed form expression of the Normalized Google Distance is:

$$sNGD(x,y) = \frac{\max\{\log n_x, \log n_y\} - \log n_{x,y}}{\log N - \min\{\log n_x, \log n_y\}} \qquad \text{(Eq. 3)}$$

where n_x and n_y are the number of results returned by a search in an online database (e.g. Google) for each individual term, N is a large number representing the total number of possible results, and $n_{x,y}$ is the number of results returned by a joint search for both terms. The main thrust the research in [Cilibrasi & Vitanyi 2007] was to develop a new theory of semantic distance between a pair of objects based on a backend derived from a set of documents.

Graphically, cosine similarity and symmetric NGD can be imagined as creating undirected edges between nodes, as shown in Figure 6, thus creating an undirected graph.

---

[7] Although the word "Google" is used, the "Google" database is not actually used in our version of the similarity metric. The use of "Google" in the similarity metric's name is used because the original authors of the paper where the algorithm was initially presented [Cilibrasi & Vitanyi 2007] used Google in their analysis and accordingly named the distance metric they created.

Dijsktra-Jarnik-Prim's (DJP) algorithm, Kruskals algorithm and Heymann algorithm are all algorithms that use undirected graphs as initial input.
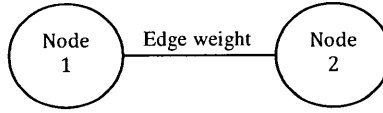
Figure 6: Illustration of Undirected Edge

3.3.3 Asymmetric Normalized Google Distance Similarity

The asymmetric NGD metric was developed in [Woon & Madnick 2008] by analyzing the symmetric NGD metric and noticing that the original closed form equation as seen in Eq. 3 uses the "max" and "min" functions and ultimately derives a symmetric relationship between two terms. The use of these functions hides some information. The idea was to create a similarity metric between a pair of terms that clearly indicates the parent-child relationship between them. As such, an asymmetric metric was needed. It was noticed that the symmetric NGD metric could be easily turned into an asymmetric measure by removing the "max" and "min" operators. In Eq. 3, the first term in the numerator and the last term in the denominator are picking the max and min of the terms n_x and n_y respectively. Since the max and min are mutually exclusive, what this equation really is doing is placing one of n_x or n_y in the numerator and the other in the denominator, leading to two possible interpretations of the formula:

$$aNGD(?,?) = \frac{\log n_x - \log n_{x,y}}{\log N - \log n_y} \qquad \text{(Eq. 4)}$$

$$aNGD(?,?) = \frac{\log n_y - \log n_{x,y}}{\log N - \log n_x} \qquad \text{(Eq. 5)}$$

To see which of the two equations above refers to term x being a child of term y, a simple test case was run. Using Engineering Village, we ran a query for "power" and "hydroelectric power". Intuitively, "hydroelectric power" should be a subset of "power", and hence should be its child.

Running a query for ["hydroelectric power"] in Engineering Village produces 30,918 results. Running a query for ["power"] produces 2,616,414 results. Running a query for ["power" AND "hydroelectric power"] displays 30,918 results. N was chosen to 10^{10}, an arbitrarily large number. From this:

$$aNGD(?,?) = \frac{\log n_{\text{"hydroelectric power"}} - \log n_{\text{both}}}{\log N - \log n_{\text{power}}} = 0 \qquad \text{(Eq. 6)}$$

$$aNGD(?,?) = \frac{\log n_{\text{power}} - \log n_{\text{both}}}{\log N - \log n_{\text{"hydroelectric power"}}} = 0.34983 \qquad \text{(Eq. 7)}$$

Since "hydroelectric power" should be a child of "power", this led to the asymmetric NGD metric:

$$aNGD(x,y) = \frac{\log n_x - \log n_{x,y}}{\log N - \log n_y} \qquad \text{(Eq. 8)}$$

where aNGD(x,y) is the aNGD associated with term x being a child of term y, with NGD ranging from 0 to 1, and the lower NGD values represent the more likely to be correct term connection. In the example above:

aNGD("hydroelectric power", "power") < aNGD("power", "hydroelectric power")

which means that it is more likely that "hydroelectric power" is a child of "power", which makes sense.

A key difference of the symmetric / asymmetric NGD similarity and cosine similarity is that a link is optimal in the NGD case if it minimizes its value, whereas in cosine similarity a larger value is seen as an indicator of a closer relationship between terms.

Graphically, the asymmetric NGD similarity can be imagined as creating directed edges between nodes, as shown in Figure 7, producing directed graphs. Edmond's algorithm and Heymann's algorithm are the two taxonomy generation algorithms that we use that require directed graphs as initial input.
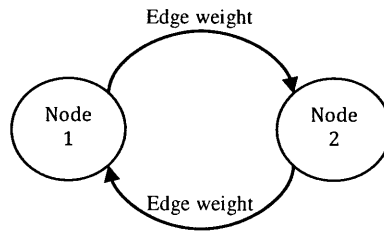
Figure 7: Illustration of Directed Edges

3.4 Populating the Term Similarity Matrix

With a similarity measure in place, the next step we take is encapsulate all the pairwise term similarity values into a form which we refer to as a *distance matrix*. A *distance matrix* is

28

shown in Figure 8, and can be imagined as a physical representation of the directed / undirected graph formed using one particular similarity metric as basis. The entry in row i, column j of the distance matrix, henceforth referred to as [i,j], represents the term similarity metric value associated for term i being a child of term j. A matrix generated using cosine similarity or symmetric NGD similarity will be symmetric across the diagonal, whereas one generated using asymmetric NGD similarity will not.

Visualizing this matrix as a graph makes it possible for us to use graph theory algorithms to process the information in the matrix to produce a taxonomy.
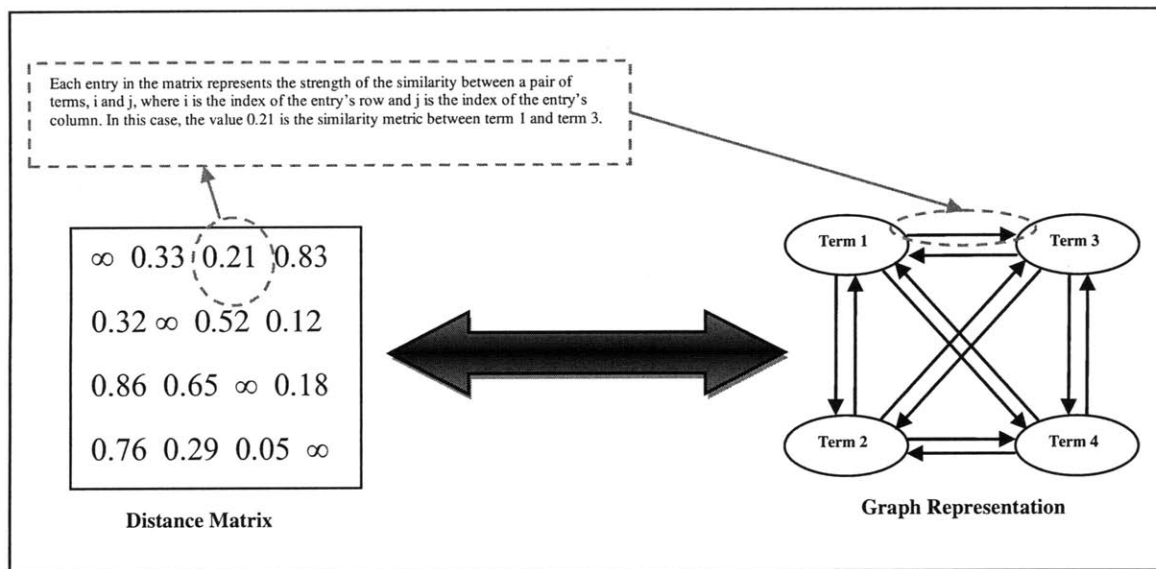
Figure 8: Representations of a Distance Matrix
A distance matrix is represented as a matrix of pairwise similarity values but can also be thought of as a graph of interconnected nodes, with each node representing a term. Notice that the diagonals of the matrix are set to ∞, this is because a term cannot be related to itself. The matrix shown above is asymmetric, indicating that the asymmetric NGD similarity metric was used.

The distance matrix is always a square matrix of fixed size dependent on the number of terms in the taxonomy. Specifically, each distance matrix is an n x n matrix where 'n' represents the number of terms in the taxonomy. Once the dimensions of the distance matrix are set, we populate the matrix with similarity values based on one of the term similarity metrics discussed earlier.

Choosing which 'n' terms to insert in the matrix is tricky. We have decided to do this by choosing the 'n' most frequently occurring terms in the bibliometric data set we generated earlier. Specifically, the number of terms, 'n', to be included in the taxonomy and the field in the data set schema from which to take the terms from should first be provided. The field from the data set schema where the terms are taken could either be the "controlled terms", "uncontrolled terms", or

"joint terms" fields in the case of an Engineering Village-collected data set, and either "author keywords", "index keywords", or both in the case of a Scopus-collected data set.

The field of interest is then scanned, and a dictionary[8] of terms is formed with a stemmed version of the term as the 'key' and the number of occurrences among all the articles as its 'value'. The top terms whose stemmed versions have the highest values are then used as the terms to be included in the taxonomy. The use of word 'stemming' is a very important step in the term collection process. "Stemming" is the process of reducing words to their stem, base, or root form. The stem need not be identical to the morphological root of the word, but it is important that similar terms map to the same stem, to eliminate redundancy. For example, the terms 'energizing' and 'energized' both stem to form the term 'energ'. Stemming the terms that appear in the field of interest first is important so that no duplicates of terms are ever found in the final term list. Without stemming, we could have a "renewable energy" taxonomy that contains the terms "energy", "energized" and "energizing" all as separate interconnected nodes. While we understand that these terms have slightly different meanings, we believe that for the most part all these terms will belong to the same general technological concept within the technological research field we are analyzing, hence it is acceptable to merge these terms together into a single term concept.

After the terms are selected, each term is mapped to a row/column in the distance matrix. Distance matrix entries can then be populated based on the term similarity metric discussed earlier. Values along the diagonal of the distance matrix are then set to undefined values since these values refer to the weight of term's link to itself, which is not allowable in the final taxonomy and hence should not exist in the graph.

The distance matrix is then used as input to one of the taxonomy generating algorithms. Each algorithm basically takes the graph representation of term similarities, as represented by the distance matrix, and turns it into a directed spanning tree. In graph theory, a directed spanning tree of a graph G is a graph containing all the vertices of G where every vertex has only one parent. In other words, the graph has no cycles. Figure 9 shows a visualization of this, where an initial graph containing 9 terms is transformed into a taxonomy by selecting edges in the graph representation of the term's similarities.

---

[8] 'dictionary' is a computer science term referring to an abstract data type composed of a collection of unique keys/identifiers, each associated with a collection of values. The operation of finding a value associated with a key is called a 'lookup' and happens in constant time.
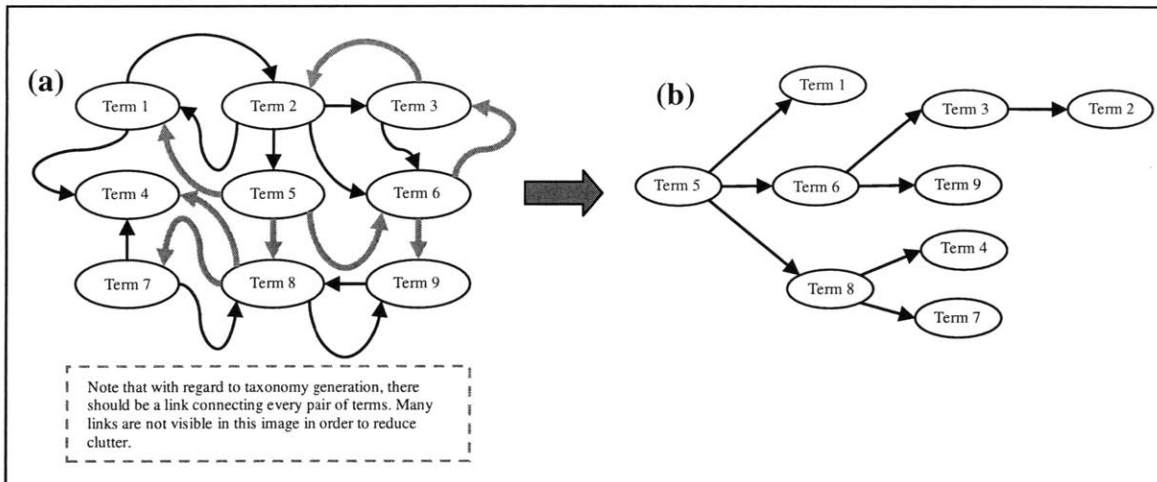
Figure 9: Transformation of Graph Representation of Term Similarity Relationships into Final Taxonomy.

The example above illustrates roughly how a taxonomy is formed based on the graph representation of its term relationships. The term relationships can be visualized as a graph like in (a) and a spanning tree is then selected from among the edges in the graph. The spanning tree is then transformed into a taxonomy by instantiating one term, in this case term 5, as the root node then forming the rest of the taxonomy by staying consistent with the connections in the spanning tree, thus creating a final taxonomy as shown in (b)

3.5 Choosing a Root Node

As a next step, many of the algorithms we use require the instantiation of a root term / root node as part of the process. We do this either by explicitly specifying a root, or by using term centrality measures to determine the most central, and thus 'root' term.

A root term can be specified manually. For example, if we have the terms ["energy", "solar energy", "wind power", "hydroelectricity"], it makes sense to specify "energy" as the root term as it is the most general concept. In practice however, term lists aren't always short enough to allow for manual inspection. Many of the taxonomies we generate will have up to thousands of terms. In these cases, manual instantiations of root terms is not advisable. In such cases, the root term should be specified by choosing the most "central" term in the graph. There are two graph centrality algorithms that we currently use: betweenness centrality and closeness centrality.

3.5.1 Betweenness Centrality

To calculate the betweenness centrality, the full list of vertex-pairs in a graph is collected and the shortest distance between each pair is found. The centrality of a vertex is then based on the number of shortest paths that pass through it. Vertices that occur on many shortest paths between nodes will have a higher betweenness centrality metric than those that do not. In other

words, betweenness centrality for a given node is the fraction of the number of shortest paths that pass through that node. The formula for betweenness centrality is:

$$C_{\text{betweenness}}(v) = \sum_{v \neq s \neq t \in V} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

(Eq. 9)

where $\sigma_{s,t}$ is the number of shortest paths from node s to node t, and $\sigma_{s,t}(v)$ is the number of shortest paths from node s to node t that pass through node v.

3.5.2 Closeness Centrality

In graph theory, closeness centrality looks at all the shortest paths between pairs of vertices, and quantifies the centrality of a vertex based on the mean of all the shortest path lengths that originate from it. Vertices that have smaller mean shortest path lengths will have a higher closeness centrality metric that those with larger values. The formula for closeness centrality is:

$$C_{\text{closeness}}(v) = \frac{1}{\sum_{\substack{t \in V \\ t \neq v}} d_{v,t}}$$

(Eq. 10)

where $d_{v,t}$ is the length of the shortest distance in the graph between vertices v and t.

32

3.6 Taxonomy Generation Algorithms

Now that the distance matrix has been populated and a root node has been chosen, a specific taxonomy generation algorithm can now be used to produce the final taxonomy. This section discusses each taxonomy generation algorithm that we use in detail. The algorithms analyzed are:

- Dijsktra-Jarnik-Prim's (DJP) Algorithm

- Kruskal's Algorithm

- Edmond's Algorithm

- Heymann Algorithm

- The Genetic Algorithm

It must be noted that with the exception of the Genetic algorithm, the other algorithms mentioned above all run to completion in about the same amount of time. The bottleneck in our implementation of the process lies in the bibliometric data set collection and the distance matrix creation.

3.6.1 Dijsktra-Jarnik-Prim Algorithm

The Dijsktra-Jarnik-Prim (DJP) algorithm finds an optimal spanning tree for a connected, undirected, weighted graph. Since DJP requires an undirected graph, it uses either the cosine or the symmetric NGD similarity distance metric. DJP is an example of a a greedy algorithm, which means that it solves a problem by making a series of locally optimal decisions, with the hope of converging on the global optimum (or a close approximation) within a reasonable amount of time.

For taxonomy generation, an adapted version of the original DJP algorithm is used:

- Determine the root term of the taxonomy

- Make the root term node the starting point of the taxonomy

- Iteratively insert the terms not yet in the taxonomy by inspecting the original graph and checking to see which term has the closest relationship with one of the nodes / terms already in the taxonomy. Do this until all the nodes / terms are in the final taxonomy

Figure 10 illustrates the DJP algorithm.
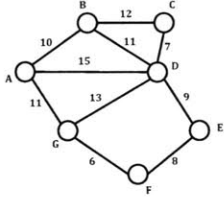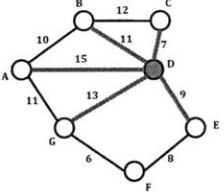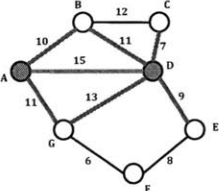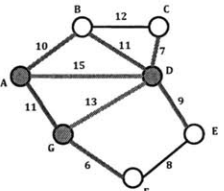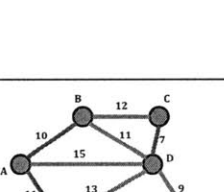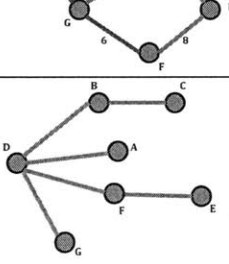
	The initial graph. For taxonomy generation purposes, this graph would be represented by a distance matrix and would be much denser than the graph on the left. The nodes in the graph represent the terms in the distance matrix and the numbers next to the links represent the similarity values between terms.
	D is chosen as the root node. This means the possible next nodes to add to the taxonomy are those directly connected to **D**: **A, B, C, E** and **G**.
	A is chosen to be included next in the taxonomy as a child of **D** as it is the best-connected node to **D** since it has the highest similarity value. Including **A** in the taxonomy now allows the inclusion of nodes **B** and **G** as children of **A.**
	G is chosen to be included next in the taxonomy as a child of **D** as it is the next best-connected node to the current nodes in the taxonomy, **A** and **D**. Including **G** in the taxonomy now allows the inclusion node **F** as a child of **G**. Including **G** also prohibits the inclusion of the arc **AG** in the taxonomy since **A** and **G** are already in the taxonomy.
	The algorithm continues until all the nodes are included in the taxonomy. The visual on the left shows the spanning tree that the final taxonomy is based on.
	The final taxonomy is produced by pulling out **D** as the root and preserving the links in the graph. The visual on the left shows the final taxonomy.

Figure 10: Illustration of DJP Algorithm for Taxonomy Generation

3.6.2 Kruskal's Algorithm

Kruskal's algorithm is another algorithm which finds the optimal spanning tree for a connected undirected weighted graph.

The general algorithm of Kruskal's is as follows: Firstly, an edge-less graph containing all vertices in the original graph is instantiated; the edges with the highest similarity values from the original graph are added sequentially into this graph until it becomes a spanning tree (similar to DJP, Kruskal's algorithm is another example of a greedy algorithm).

As Kruskal's algorithm also uses undirected graphs, the cosine similarity or symmetric NGD metrics are used. In the context of taxonomy generation, the following customized version of the algorithm was used:

- Create an initial set of individual, unconnected nodes, collectively called **S**, where each node represents a term in the undirected distance matrix generated using either the cosine similarity metric or the symmetric NGD similarity metric as basis

- While **S** is not fully connected[9]:

  o Take the edge *e*, with the best edge-weight value in the distance matrix

  o Add edge *e* to **S** if the *e* does not create any cycles in **S**, otherwise discard the edge.

- Transform **S** into a directed spanning tree by instantiating a root node and creating directionality by staying consistent with the connections in **S**. This last step is the only new step added in order to adapt Kruskal's algorithm for taxonomy generation.

Kruskal's algorithm is deterministic. This means that it only produces one result for a given distance matrix. Figure 11 illustrates the algorithm.

	The initial graph. For taxonomy generation purposes, this graph would be represented by a distance matrix and would be much denser than the graph on the left. The nodes in the graph represent the terms in the distance matrix and the numbers next to the links represent the similarity values between terms. Term_Clusters = []

---
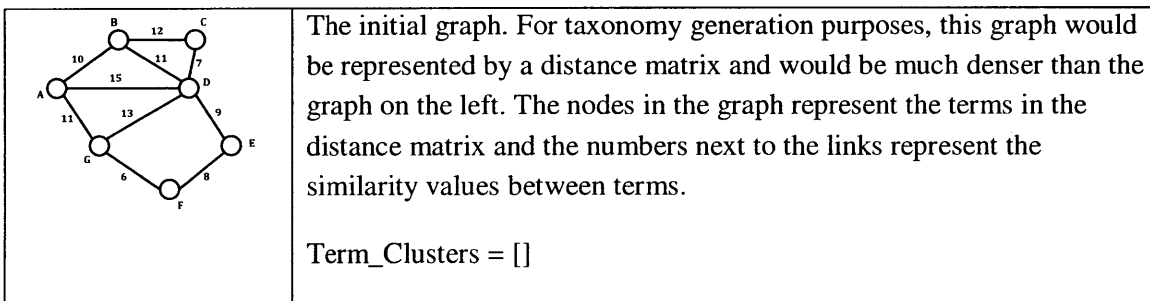
[9] A graph is fully connected when there is a traceable path between every pair of vertices in the graph
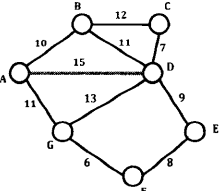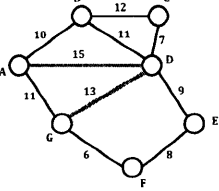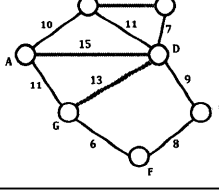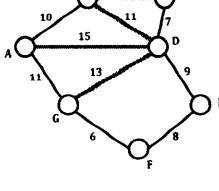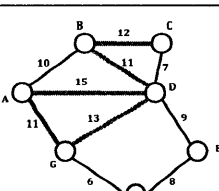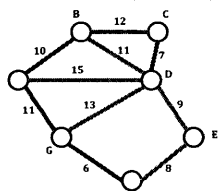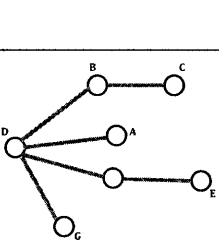
	The longest (best) arc is **AD** with length 15, hence it is included in the final spanning tree. Term_Clusters = [[A,D]]
	The next best arc is **DG** with length 13, hence it is included in the final spanning tree. Term_Clusters = [[A,D,G]]
	The next best arc is **BC** with length 12, hence it is included in the final spanning tree. Since **BC** is not connected to the [A,D,G] term cluster, a new cluster is created. Term_Clusters = [[A,D,G],[B,C]]
	The next best arc is **BD** and **AG** with length 11. **BD** is randomly selected to be added to the spanning tree first between the two. Since the **BD** arc connects the [B,C] and [A,D,G] clusters, these two clusters are merged. Term_Clusters = [[A,B,C,D,G]]
	The next best arc is **AG** with length 11. However, since it connects two nodes that are already in the same cluster, the edge is discarded. Term_Clusters = [[A,B,C,D,G]]
	The process continues until all the arcs are accounted for and the spanning tree is connected. For taxonomy generation, the initial graph is one where each node is connected to every other node, hence it is guaranteed that at the end of the algorithm's run, all the nodes will be connected and a spanning tree will be formed. The final spanning tree is then converted to a taxonomy by instantiating a root node / term.
	The final taxonomy is generated by choosing a root node, and pulling it out while preserving the links in the graph. The visual on the left shows the final taxonomy produced by choosing **D** as the root node.

Figure 11: Illustration of Kruskal's algorithm for Taxonomy Generation

3.6.3 Edmond's Algorithm

Edmond's algorithm is a graph theoretic algorithm for finding a minimum spanning tree. Unlike the previous two algorithms discussed in this section, Edmond's uses a directed, weighted graph as input, hence the asymmetric NGD (aNGD) similarity metric is used.

The algorithm is as follows:

- Create an initial graph, called **S**, where every node represents a term in the distance matrix and each node only has one incoming edge, which is the incoming edge with the minimum (and thus best) aNGD similarity value
- Choose a root node and remove its incoming edge
- For each cycle in **S**, remove an existing edge and add an outside edge (thus removing the cycle) by first calculating the minimum *additional cost* value for each node currently in the cycle, where the *additional cost* is defined as:

$$additional\_cost(v) = \min_{\substack{w \in V \\ !(w \in loopNodes) \\ [u,v] \in loopEdges}} (edge[w \to v] - edge[u \to v]) \qquad \text{(Eq. 11)}$$

where edge[u → v] represents the aNGD similarity metric value (found in the distance matrix) for the connection (u → v). Note since each node only has one incoming edge, the edge (u→v) is the only edge incoming to node v in the loop.

The *additional cost* metric for a particular node in a cycle can be thought of as the minimum incremental amount that will be added to the total path length of the cycle if one if its edges is replaced by a different edge coming from outside the cycle.

- For each cycle in S, remove the edge (u→v) and add the edge (w→v) where v is the node in **S** that produced the minimum *additional cost*, and (w→v) is the incoming edge to node v from outside the cycle that produces node v's minumum *additional cost*.

A taxonomy cannot have any cycles in it. Edmond's algorithm basically analyzes the taxonomy and breaks each cycle until a valid taxonomy is formed. Figure 12 illustrates the process of *additional cost* computation and cycle fixing.
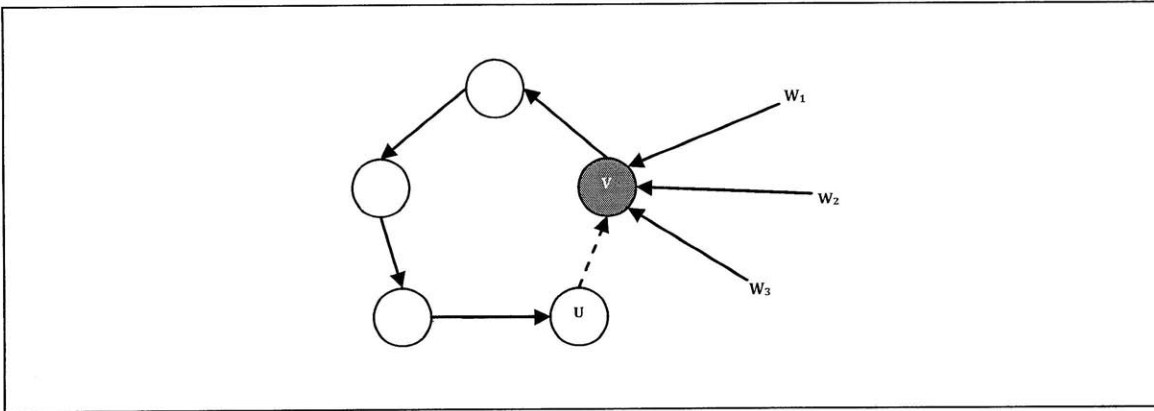
Figure 12: Cycle Fixing Process in Edmond's Algorithm.
To fix / break cycles, Edmond's algorithm analyzes each node in the cycle and computes its *additional cost* by computing what is the minimum incremental addition to the total path length that brought about by adding one of its incoming edges from nodes outside the cycle (in the figure above, w_1, w_2, or w_3) and removing its incoming edge from within the cycle (in the figure above, u→v). The node with the minimum 'additional cost' has its in-cycle incoming edge removed and its minimum-valued outside-cycle incoming edge added.

Figure 13 illustrates the algorithm.

	The initial graph. For taxonomy generation purposes, this graph would be represented by a distance matrix and would be much denser than the graph on the left as every node would have to be connected to every other node. The nodes in the graph represent the terms in the distance matrix.
	Edges are removed from the graph such that there is only one incoming edge for every node. In other words, only the most optimal incoming edge for each node is kept in the graph.
	After the root node is chosen, the incoming edge to the root node is removed. In this case, **D** was chosen as the root node and hence the edge **CD** was removed from the graph, producing the visual on the left.

38

	Cycles are detected one-by-one and fixed accordingly. In this case, the cycle **ABG** is detected.
	The detected cycle is fixed based on the procedure described in this section and illustrated in Figure 12. In this example, **BG** was removed and **DG** was added to the graph.
	No more cycles are detected, hence the final taxonomy can be formed by pulling out the root node, **D**, and forming the rest of the taxonomy by preserving the links in the graph. The visual on the left shows the final taxonomy.

Figure 13: Illustration of Edmond's Algorithm for Taxonomy Generation

3.6.4 The Heymann Algorithm

The work in [Heymann 2006] focused on taxonomy generation originally intended for social networks, where users annotate documents or images with keywords, a process known as 'tagging'. These collaborative tagging systems have become very popular over the past few years with online content in websites such as CiteULike.org, Del.Icio.Us, and Flickr. However, the tags displayed in these websites are unstructured. As such, the only way to aggregate tag data is to show what tags occur most frequently at any given point in time. Figure 14 shows an example of a tag cloud.

Figure 14: Example of a Tag Cloud

Here, the population of each of the world's countries is indicated based on the size of the text in the cloud [source: Wikipedia]. As can be seen, the cloud is composed of just a series of terms and there is no structure to the cloud.

[Heymann 2006] proposed a simple, efficient algorithm for converting a large set of tags into a navigable hierarchical ontology (another term for taxonomy) of tags. Each term (or tag) is associated with a vector that contains the annotation frequencies for all documents, which can then be compared to the vectors of the other terms using a variety of similarity measures thus producing an ontology where tags that are very similar to each other are linked together.

The pseudocode for the algorithm, taken from [Heymann 2006] is shown in Figure 15.

Require: $L_{generality}$ is a list of tags $t_i,...t_j$ in descending order of their centrality in the similarity graph.

Require: Several functions are assumed: $s(t_i, t_j)$ computes the similarity (using cosine similarity, for example) between t_i and t_j. *getVertices*(G) returns all vertices in the given graph, G.

Require: *taxThreshold* is a parameter for the threshold at which a tag becomes a child of a related parent rather than of the root.

```
Gtaxonomy ← {NULL, root}
for i = 1...|Lgenerality| do
        ti ← Lgenerality[i]
        maxCandidateVal ← 0
        for all tj ∈ getVertices(Gtaxonomy) do
                if s(ti, tj) > maxCandidateVal then
                        maxCandidateVal ← s(ti, tj)
                        maxCandidate ← tj
                end if
        end for
        if maxCandidateVal > taxThreshold then
                Gtaxonomy ← Gtaxonomy U {maxCandidate, ti}
        else
                Gtaxonomy ← Gtaxonomy U {root, ti}
        end if
end for
```

Figure 15: Heymann algorithm pseudocode taken from [Heymann 2006]

The algorithm requires a list of tags in descending order of their generality. It then obeys the order starting with the most general tag and iteratively inserts each tag into a growing taxonomy by attaching them to either the most similar tag or the taxonomy's root. There is one threshold used in the algorithm: taxThreshold, representing the value of the similarity measure above which a link is permitted to be a child of a tag other than the root.

[Henschel et al. 2009] then adapted the Heymann algorithm to general taxonomy creation by turning each of the terms in the taxonomy into a 'tag', and making several other important changes:

First, the term generality measure was derived from the term's centrality measure, betweenness or closeness, which is also described in section 3.5 of this paper. Second, intermediate re-ranking of the remaining terms with respect to their centrality after inserting a term into the taxonomy was included as an option in the algorithm. Finally, the concept of entropy was introduced, which is an information theoretical concept that can be used to quantitatively justify the decision of creating an edge between nodes. The entropy measures the similarity of the node to be inserted to the other nodes that are already in the taxonomy. The closer the entropy of each node is to zero, the more "accurate" its link to its parent is said to be.

In order to test the accuracy of the algorithm, [Henschel et al. 2009] used a benchmark taxonomy called MeSH (Medical Subject Headings), a manually curated ontology for medical terms. It then compared the relationships between terms in the MeSH taxonomy with the relationships formed in a taxonomy with these same terms generated using the Heymann algorithm. Overall, [Henschel et al. 2009] showed that:

1. Among the centrality algorithms, unweighted betweenness centrality generally performs best but often only marginally better than the faster unweighted closeness centrality

2. The best taxonomies generated using the closeness and betweenness centrality measures are not identical

3. Weighted similarity graphs rarely improved the performance and hence did not justify the higher computational cost

4. Re-ranking the centrality often improves algorithm performance but increases the computational expense

5. Entropy-based filtering creates more precise, but less complete taxonomies

The version of the Heymann algorithm used in this project is similar to the version presented in [Henschel et al. 2009], which is based on the work in [Heymann 2006]. The algorithm is as follows:

*   Create a set of tags, each representing a term to be inserted in the taxonomy
*   Sort all the tags by centrality. This is done by first modeling the problem as a graph where the vertices are the tags and the edges are weighted by the tags' similarity (cosine, symmetric NGD, or asymmetric NGD) relationship to each other, then applying either betweenness or closeness centrality to rank the terms by order of centrality in the graph.
*   Take the most central tag and insert it in the taxonomy.
*   Iteratively sort the remaining tags not yet in the taxonomy by centrality (optional), take the most central tag among these tags, and decide whether to insert it into the taxonomy or throw it out completely. Do this until all the tags are accounted for.

Although the general algorithm is the same, it must be noted that the algorithm used in this thesis is slightly different from the one presented in [Henschel et al. 2009]. The differences are:

- In this thesis, the use of symmetric / asymmetric NGD similarity as well as cosine similarity was allowed as a relationship metric. In [Henschel et al. 2009], only cosine similarity was used.
- In this thesis, the overall entropy of the system was not taken into account, whereas in [Henschel et al. 2009] it was prioritized.
- In this thesis, only taxonomies with one root were created. [Henschel et al. 2009] allowed for the creation of taxonomies with multiple roots, which we decided against, since we believe that a taxonomy with two root nodes is essentially two separate taxonomies. In addition, since all the other taxonomy generation algorithms we use produce taxonomies with only one root, we believe that staying consistent with the "one root" implementation made the most sense.

The Heymann algorithm is unique from the other taxonomy generation algorithms considered because it is the only one that discards terms it deems irrelevant. In our implementation, there is a threshold that can be set, t_s, that represents the similarity value beyond which a link is not allowed to exist in a taxonomy. At each iteration of the Heymann algorithm, the similarity value of the most central remaining term with the most closely related node in the taxonomy must be below this threshold, otherwise it is thrown out. Figure 16 illustrates the Heymann algorithm.

In this example, Term C is the next term to be inserted into the taxonomy as Term A and B are already in the final taxonomy.

Check to see which term currently in the taxonomy in the taxonomy target term is most similar to by inspecting the corresponding row/column in the distance matrix, and insert the term in the taxonomy accordingly. In this case, Term C was most similar to Term B among all the terms currently in the taxonomy (Terms A and B), hence it was inserted as a child of Term B.

Figure 16: Illustration of the Heymann Algorithm for Taxonomy Generation

The Heymann Algorithm inserts new terms into the taxonomy by first sorting the terms not yet in the taxonomy by centrality, then taking the most central term not yet in the taxonomy and comparing it to each of the terms in the taxonomy to see which one it is most similar to.

3.6.5 The Genetic Algorithm

The Genetic algorithm is a search technique used in computing to find solutions to optimization / search problems by means of repeated iterations, mutations and crossovers. It is typically implemented as a computer simulation in which a group of abstract representations are optimized repeatedly, generating a better set of representations at each succeeding iteration.

The previous taxonomy generation algorithms discussed in this section have all been deterministic, graph-theory based algorithms. The Genetic algorithm is neither deterministic nor graph theory-grounded, but it is still directly applicable for taxonomy generation. However, because of its non-deterministic (random) nature, it is impossible to test the validity of the Genetic algorithm as a taxonomy generation algorithm since all its outputs are inherently randomly generated so it is possible and highly likely that repeated runs of the algorithm even given exactly the same parameters will yield vastly different results.

43

In this thesis, the general Genetic algorithm is applied for taxonomy generation as follows:

- Choose an initial set of randomly generated taxonomies, each having the same root term.
- Iterate several times. For each iteration:
  - Select random taxonomies in the set, then modify ('mutate') and recombine ('crossover') these taxonomies to create the potential new set of taxonomies for this iteration.
  - Compute the score of each potential taxonomy in the set based on the weights of the edges contained in the taxonomy (which can be taken from the distance matrix). There are several ways of computing the "score" for a taxonomy which we discuss in the latter part of this thesis.
  - Select the top scoring taxonomies in the set, and discard the rest.
  - Duplicate the top scoring taxonomies so that the total number of taxonomies in the set is the same as the initial condition.

The modification ('mutation') and recombination ('crossover') process in the implementation of the GA relies on random transformations of taxonomies. A *mutation* of a taxonomy moves a random subtree of a taxonomy to a new location in the same taxonomy. A *crossover* of taxonomies transplants a random subtree from one taxonomy into a random location in another. Doing this creates duplicate nodes in the transplanted taxonomy. To fix this, the transplantation stage is immediately followed by a repair stage, where the original nodes that are now duplicated in the transplanted taxonomy are removed and the descendants of these nodes are promoted to the ancestor nodes of the original invalidated nodes. An illustration of the mutation and crossover process is shown in Figure 17.
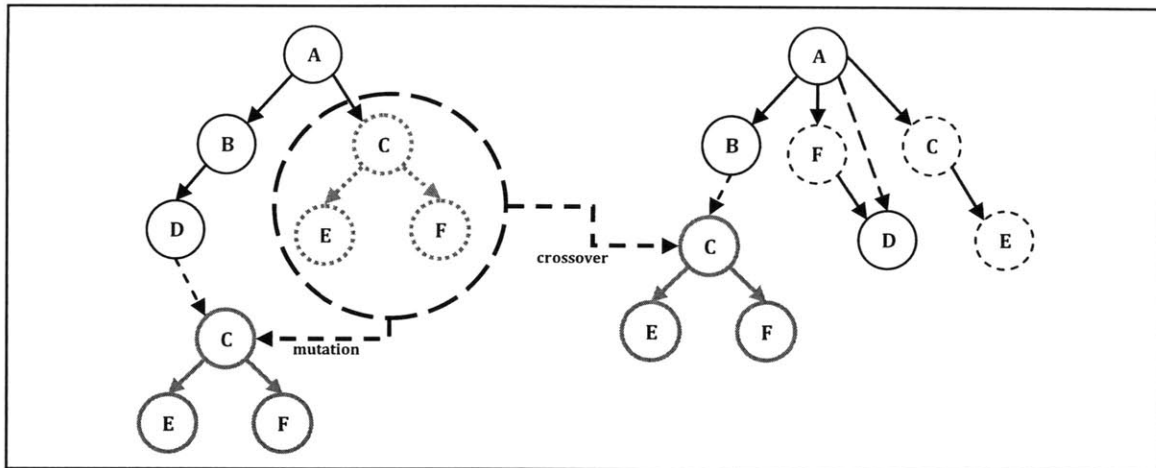
Figure 17: Mutation and Crossover Process in the Genetic Algorithm

A mutation is shown in the taxonomy on the left, where the subtree containing nodes [C,E,F] is moved into a random location on the same tree. A crossover is shown from the taxonomy on the left to the taxonomy on the right, where the subtree containing nodes [C,E,F] on the left is moved into a random location in the taxonomy on the right. Moving these nodes creates duplicates of nodes [C,E,F] in the taxonomy on the right, hence the crossover phase is followed by a repair phase where the original nodes [C,E,F] on the right are removed from the taxonomy and their children are promoted as children of their ancestors. In this case, D is promoted to be a child of A following the removal of C.

Each iteration of the Genetic algorithm relied on scoring each of the randomly modified / recombined taxonomies using a cost / scoring function. For this, we created a suite of methods that take each term's direct / indirect link's edge weight (indicated by its corresponding value in the distance matrix) in the taxonomy, and sum them up in a weighted fashion to produce a final score. These methods are discussed in the next chapter of this thesis in the subsection discussing the evaluation of individual taxonomies.

There are several customizable parameters in the Genetic algorithm, which are:

- The root node / term of the taxonomy
- Number of iterations of the algorithm
- Number of initial taxonomies
- Number of mutations
- Number of crossovers
- Number of top scoring taxonomies to keep for the next iteration
- The cost function formula to score taxonomies. The different formulas we considered are discussed in the latter section on 'Evaluating Individual Taxonomies'

The stochastic nature of Genetic algorithms means that a different final result is possible each time the algorithm is run. Specifically, this is caused by the fact that the algorithm starts with an initially generated set of random taxonomies, where terms are connected to other terms randomly. After this, random changes are applied to the taxonomies to generate the new set of taxonomies for the next iteration. At each iteration, more random transformations are then performed until the end of the algorithm. Hence, even if given the same initial parameters, it is rare to expect the Genetic algorithm to produce the same final taxonomy as the size of the taxonomy gets larger as most of the process is stochastic. This random nature of the algorithm makes it difficult to evaluate its effectiveness in relation to the other taxonomy generation algorithms presented in this thesis, however the Genetic algorithm does provide a very flexible framework in which a variety of different cost functions can be easily tested without having to devise new optimization algorithms. Although the Genetic algorithm is not tested in this thesis, it is still a usable taxonomy generation algorithm hence we devoted a section to it here.

3.7 Viewing Taxonomies

After processing a distance matrix using one of the taxonomy generation algorithms, the result is a taxonomy. Visually, every taxonomy can be imagined as a directed spanning tree[10]. An example of a taxonomy is shown in Figure 18. The taxonomies typically generated in this thesis are large in scale, and hence the proper presentation of the taxonomy to the end-user / viewer is essential. The viewer may simply want to see the overall shape of the taxonomy, or at other times may want to zoom into specific sections of the taxonomy to find particular terms or links between terms. The graphical user interface for viewing taxonomies must allow for any form of user manipulation.
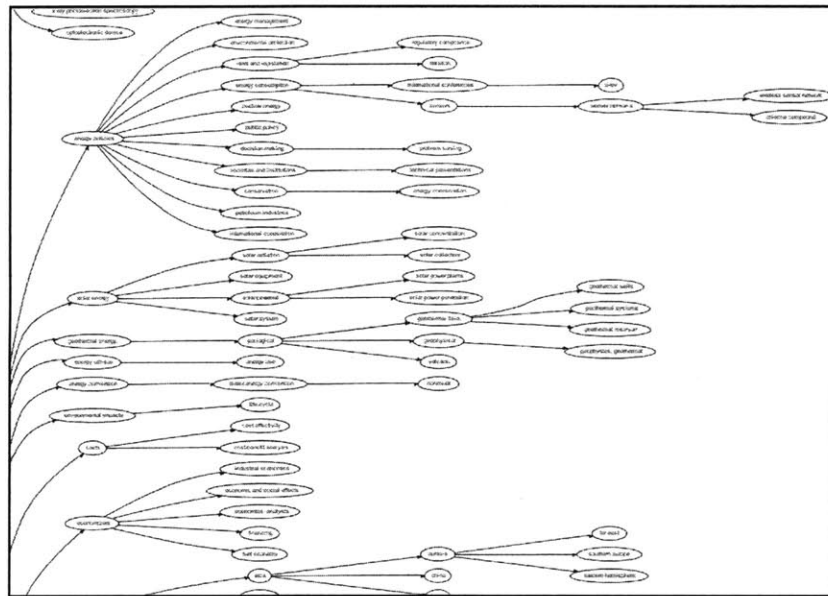
Figure 18: A cross-section of the visual representation of the 500-term "renewable energy" taxonomy using the Heymann algorithm, cosine similarity, closeness centrality

To allow the taxonomies to be easily visualized, ZGRViewer, an open source application, is used. The interface of ZGRViewer is very simple and learnable. A screenshot is shown in Figure 19. ZGRViewer provides the option to the user to view the overall shape of the taxonomy, zoom into specific sections, navigate around the taxonomy's structure, and find specific terms by typing the term's name into a search box.

---

[10] In graph theory, a spanning tree of a graph G is a graph composed of all of the vertices of G and a subset of its edges such that there are no cycles in the graph.
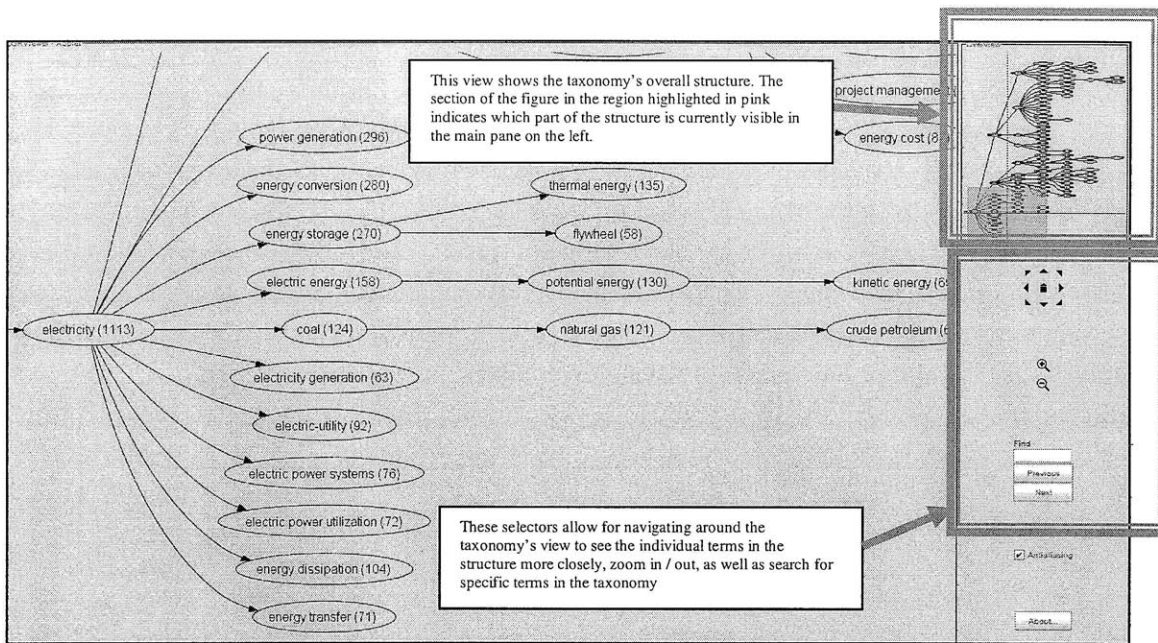
Figure 19: The ZGRViewer Interface
ZGRViewer provides a view of the overall shape of the taxonomy in the upper righthand corner, with a pink section highlighter indicating which section of the taxonomy is currently being viewed in the main window on the left. In the lower righthand side, there are controls that allow the viewer to zoom in, zoom out, move through the document, and search for specific text within it.

Every term in the taxonomy is linked to one or more other terms via a directed edge. Every directed edge carries a corresponding edge weight whose value can be obtained by referencing the taxonomy's distance matrix.

In our code, taxonomies are represented in two ways:

1. A sparse connection matrix. This matrix contains only '1's and '0's, with a '1' value representing a link. If the value in index [i,j] of the matrix is a '1', then term i and term j are said to be connected in the taxonomy with term j as the parent of term i. If the value in [i,j] and [j,i] are '0's, then terms i and j are not connected.

2. A "tree" object. A tree is represented by an organized collection of node objects where each node represents a term in the taxonomy. Aside from the root, every node contains a pointer to its parent node. In addition, nodes contain pointers to each of their child nodes.

In the software developed in this thesis, the final taxonomies can be visualized by the user in two forms:

1. As a text file

2. As an SVG (scalable vector graphics) file, which is an xml-based file format that can represent two-dimensional vector graphics in any dimension without loss of clarity. This is the required input to the ZGRViewer software.

48

3.8 Taxonomy Generation Process Summary

This chapter discussed all the steps involved in the taxonomy generation process, starting with the online data mining process continuing to the generation and visualization of the taxonomy. The process of taxonomy generation is by no means trivial, and generating a taxonomy involves several user inputs within the process. After coming up with the seed term, the user must make several choices, outlined below:

1. Choose a source for term extraction:
    a. Compendex
    b. Inspec
    c. Scopus
2. Choose how many keywords to include in the taxonomy
3. Choose the column in the article database from which to gather terms:
    a. If Compendex / Inspec:
        i. Controlled Terms
        ii. Uncontrolled Terms
        iii. Both Controlled and Uncontrolled Terms
    b. If Scopus:
        i. Author Keywords
        ii. Index Keywords
        iii. Both Author and Index Keywords
4. Choose the basis of the term similarity metric:
    a. Co-occurrence among article term / keyword lists
    b. Co-occurrence among article abstracts
    c. Co-occurrence among article titles
    d. Any combination of (a), (b), (c)
5. Choose the root node of taxonomy via:
    a. Manual input
    b. Betweenness centrality
    c. Closeness centrality
6. Choose the Taxonomy Generation Algorithm
    a. Using Distance Matrix based on Cosine Similarity or Symmetric NGD Similarity
        i. Dijsktra-Jarnik-Prim's (DJP) Algorithm
        ii. Kruskal's Algorithm
        iii. Heymann Algorithm
            1. Choose centrality algorithm (betweenness, closeness)
            2. Choose t_s threshold to discard irrelevant nodes
    b. Using Distance Matrix based on asymmetric NGD similarity
        i. Edmond's Algorithm
        ii. Heymann Algorithm
            1. Choose centrality algorithm (betweenness, closeness)
            2. Choose t_s threshold to discard irrelevant nodes

A diagram of the process is shown on Figure 20.
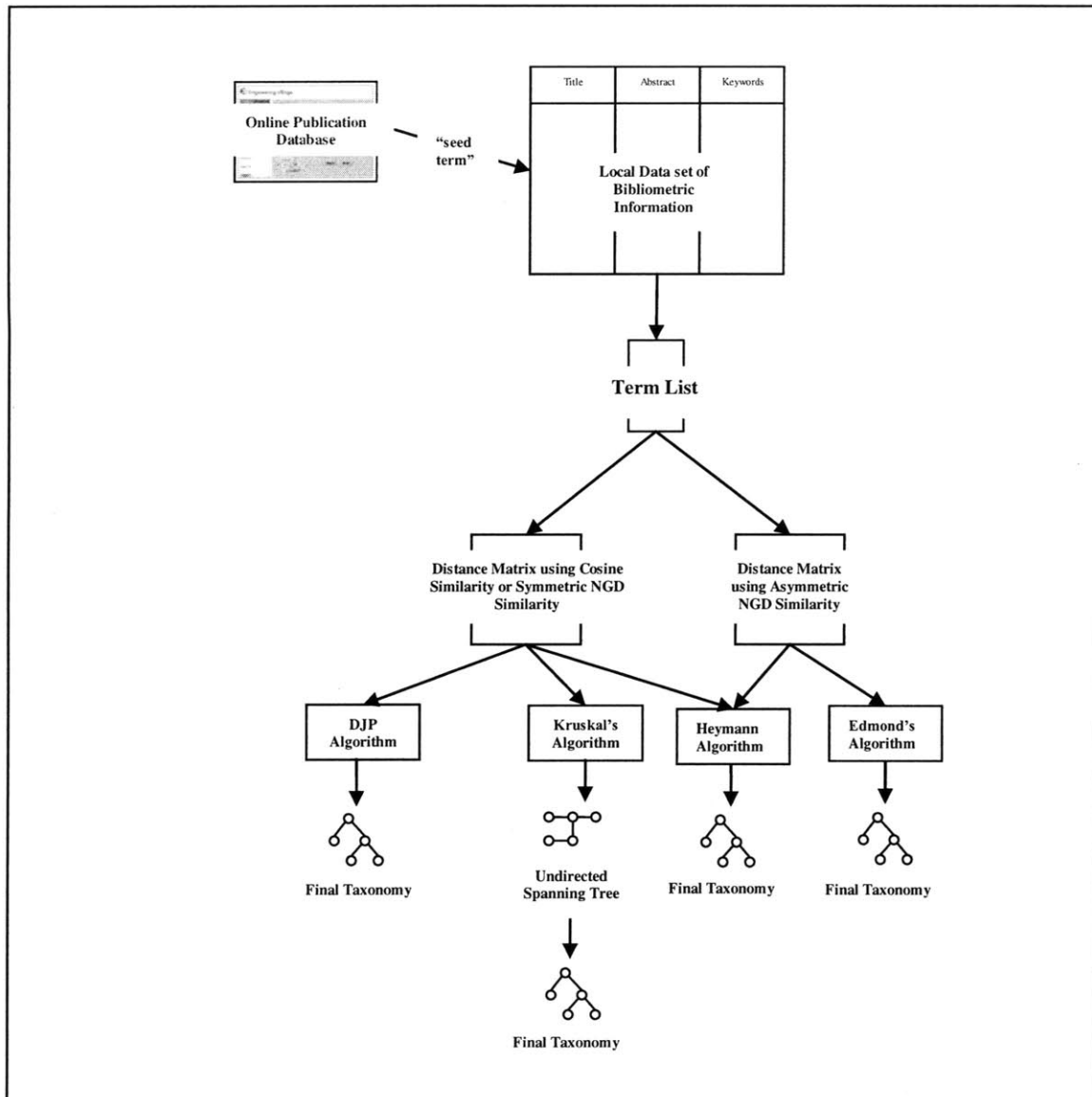
Figure 20: Diagram of the User Decision Path for Taxonomy Generation

The next chapter will discuss the methodology which we used to test the taxonomy generation algorithms described here, as well as motivate the tests by describing the underlying assumptions behind the taxonomy generation process.

CHAPTER 4: Taxonomy Evaluation Methodology

4.1 Introduction

In the last chapter, each of the taxonomy generation algorithms and the process of collecting and analyzing the bibliometric information contained in an online publication database were discussed. As was seen, there are many algorithms that can be used to produce taxonomies. All of these methods are grounded in solid mathematical theory, but the choice as to which algorithm is best is still unclear. This chapter will discuss the methodology we used to analyze each of the specific taxonomy generation algorithms.

As a starting point to our analysis, we made the intuitively reasonable assumption that for a given domain of knowledge, there exists a knowledge "landscape" representing the various concepts, relationships and subdomains which constitute the domain. Further, it can be envisaged that, given a set of related terms, it should be possible to capture the interrelationships between these components in a suitable form, such as a term relationship taxonomy. It must be noted however, that while we are assuming there exists only one true landscape, capturing this as a taxonomy is a non-unique process, and therefore there may be more than one valid, representative taxonomy. However, for the purpose of simplicity, we will be assuming a single unique taxonomy within the experiments described in this section.

Figure 21 illustrates our underlying model. We believe that every research area has several concepts that are central to the field and are interrelated in some way. However, this underlying structure is typically not observable but is manifested in the form of documents and articles written by researchers in this area. In turn, these documents and articles are collected, stored locally, and subsequently accessed and analyzed by our software in the form of bibliometric indicators. Finally, our taxonomy generation algorithms analyze the information and attempt to unearth the central ideas / terms and organize them in the form of a taxonomy that reflects the relationships between these ideas / terms.
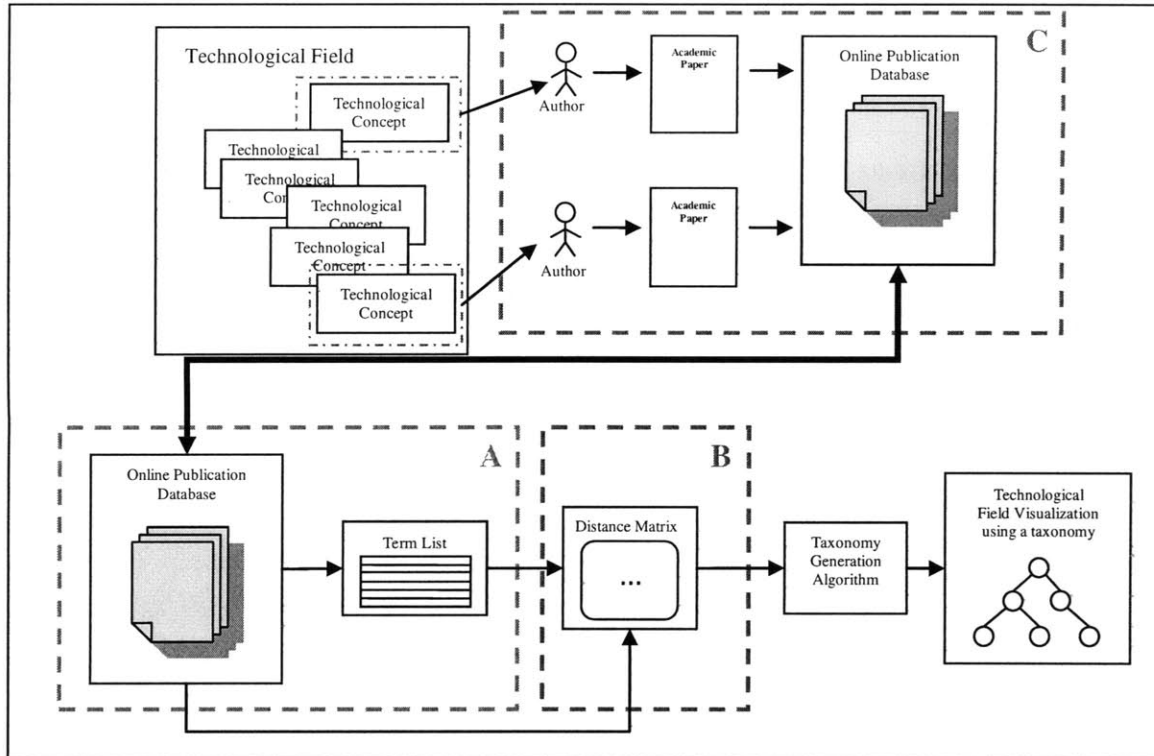
Figure 21: The underlying model behind the taxonomy generation process

Another important assumption that was made is that the bibliometric information thus gathered is imperfect, where the origins of this imperfection are twofold: firstly, there could be errors and biases in the documents, as well as non-uniform coverage of the underlying area. Secondly, it is typically not possible to collect or analyze all relevant relevant bibliometric data as this depends on the quality of the specific database used.

A further concern is that, even if we were to assume perfect data, inferring the underlying taxonomy remains a difficult challenge, and is an instance of class of problems known as an inverse problem. Solving these problems require the careful use of effective retrieval algorithms.

In view of this model of events, a set of experiments have been devised which should allow the taxonomy generation process to be appropriately tested. These will be described in the following section.

4.2 Taxonomy Evaluation Criteria

Manually examining a taxonomy gets increasingly difficult as more terms are introduced since each node becomes less visible in the taxonomy's graphical representation. The standards by which a taxonomy should be judged are also uncertain. What makes one taxonomy better than another? What is implied when one term is linked to another in a given taxonomy? What makes a

link correct / wrong? For a taxonomy with thousands of terms, who is the ultimate arbiter that decides the validity of a taxonomy?

For this thesis, we attempted to evaluate the efficacy / validity of the different taxonomy generation algorithms with the goal of deciding which algorithm works best. To automatically evaluate taxonomies and the algorithms used to generate them, we took advantage of the mathematically-grounded nature of the taxonomy generation process. It is intrinsically difficult to properly evaluate a taxonomy generation approach as there are yet to be any established standards in literature, however, based on our assumptions regarding the underlying model behind taxonomy generation we believe that a good taxonomy-generation algorithm must be one that has these three characteristics:

1. It must produce consistent taxonomies despite slight perturbations to its backend, or slight changes to the terms in the taxonomy. This is necessary given the issue of imperfect information mentioned in the previous section.

2. It must conform well to the the pairwise-relationship-strength matrix (distance matrix) which it is based on, thus maximizing the overall similarity of terms in the taxonomy. This is necessary because even in the hypothetical scenario where there is perfect data, there is still the issue of solving the "inverse problem" as mentioned in the previous section.

3. It must produce taxonomies that are valid representations of relationships between terms in the taxonomy. This is necessary because in the end, the deliverable for our team's project is a taxonomy that accurately represents the research landscape. Even if the first two characteristics for a good taxonomy are met, if the final output is a taxonomy that intuitively does not make sense, all our work is invalidated.

To test conformity to these three conditions, the following analyses were performed:

1. The consistency of each algorithm is evaluated by attempting to vary either the backend data set or the term list used in the taxonomy. Referring back to Figure 21, this can be seen as trying to perturb box A and seeing its effect on the outputted taxonomies.

2. Each taxonomy produced using each algorithm is scored based on its conformity to the distance matrix. Referring back to Figure 21, this can be seen as measuring how well the taxonomy generation algorithm can encapsulate the information in box B.

3. Synthetic data based on a predefined underlying model is generated and compared against the taxonomy generation's output to evaluate each algorithm's effectiveness. Referring back to Figure 21, this can be seen as creating our own documents / publication database, much like box C.

4. The taxonomies produced by the taxonomy generation algorithms that fare well in the first three tests are then manually analyzed to check for their intuitive validity.

4.3 Evaluating the Consistency of Taxonomy Generation Algorithms

We believe that a taxonomy generation algorithm must produce *consistent* taxonomies each time. We define *consistency* as robustness towards noise. Taxonomy generation algorithms that are *consistent* must produce similarly structured taxonomies despite slight variation to its inputs. This is an important requirement since the underlying taxonomy clearly does not change even if different perspectives of it may exist.

The first set of consistency tests were done by varying the bibliometric data set that is the basis of every taxonomy generation algorithm. We believe that every good taxonomy generation algorithm must consistently produce the same taxonomies despite slight variations to its backend data set. As mentioned in the previous subsection, we cannot assume that the data encapsulated by the backend bibliometric data set is perfect. Thus, a good taxonomy generation algorithm needs to produce similar-looking taxonomies even when the backend data set is altered slightly.

To simulate the slight variation of the backend, tests were run where the data set was varied by taking subsets of the original database. Specifically, we generated taxonomies using the same term list but took subsets of the data set. The structure of the taxonomies generated using the subset data sets were then compared to the taxonomy generated using the one that used the entire backend data set. The taxonomies must, as much as possible, contain the same links.

For example, in a "renewable energy" related taxonomy, if the terms "wind energy" and "turbines" are linked directly in the taxonomy generated using the entire backend data set, then they should also be directly linked in a taxonomy produced using only a subset of the backend data set, provided that the two taxonomies were generated using the same term lists.

The second set of consistency tests were done by varying the terms used. Taxonomies were produced using the different taxonomy generation algorithms but the terms used for each taxonomy were varied while keeping the same backend data set. Specifically, each algorithm was run with a fixed term list using the entire bibliometric data set as backend. Then, some additional terms were added to the term list, to simulate the "noise" that could be introduced in the taxonomy generation process and the entire algorithm was rerun. The outputted taxonomies in both runs were then compared to each other. The taxonomies must, as much as possible, contain the same relative term relationships.

For example, in a "renewable energy" related taxonomy, if the terms "wind energy" and "turbines" are linked directly, adding a few "noise" terms to the taxonomy and re-running the algorithm should still produce a taxonomy where "wind energy" and "turbines" are linked together, although not necessarily directly, provided that the taxonomies were generated based on the same backend data set.

For the term list consistency test, a recurring test involved comparing a small taxonomy to a larger taxonomy that contained a superset of the smaller taxonomy's terms. To do this, the larger taxonomy needed to be simplified so that its links can be directly comparable to the smaller taxonomy. This was accomplished by first creating a new root node, and instantiating it as the root of both taxonomies. Then, the terms in the larger taxonomy were scanned and the terms that did not exist in the smaller taxonomy were removed, promoting the children of the removed terms as children of existing terms. Figure 22 illustrates the process.
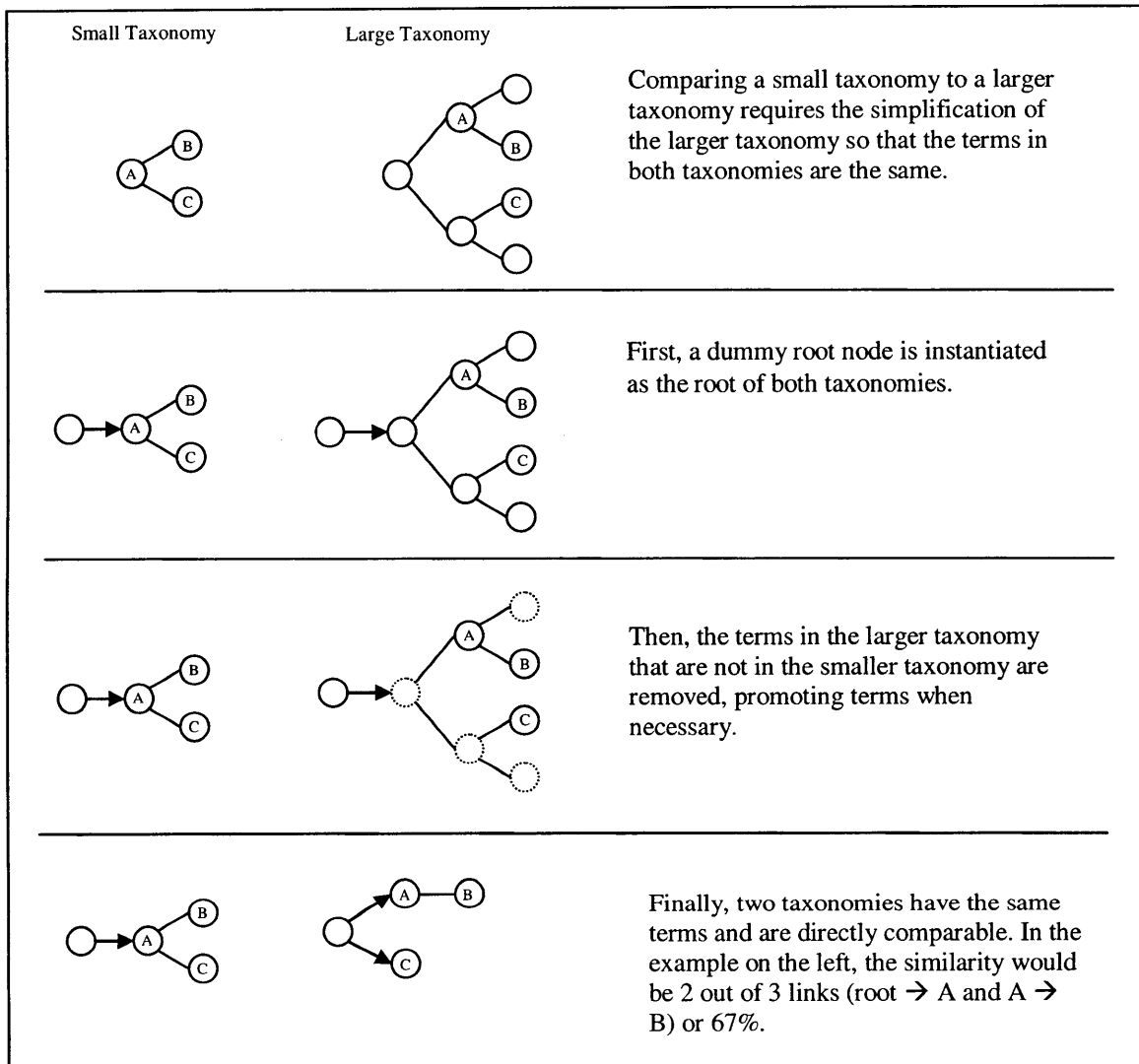
Figure 22: Simplifying a Larger Taxonomy.
The figure above illustrates the process taken to simplify a larger taxonomy so that its terms are comparable to the terms in a smaller taxonomy whose terms are a subset of the larger taxonomy's terms.

4.4 Evaluating Individual Taxonomies

After checking to see the consistency of each taxonomy generation algorithm, the next step was to individually analyze each of the taxonomies outputted by each algorithm. Within this set of tests, we chose not to manually inspect the generated taxonomies as we wanted to veer away from any sort of subjective test where results would be biased based on an individual's opinion. As such, in order to automatically verify the validity of a taxonomy generation algorithm, we decided to test the degree to which each taxonomy conformed to the pairwise-relationship-strength matrix (distance matrix) that represented the backend bibliometric data set. A numerical score was assigned to each taxonomy generated using a each different taxonomy-generation algorithm. In order to compare the different taxonomy generation algorithms to each other, taxonomies were generated using the same term list and backend data set, and they were compared using a variety of scoring metrics.

Each scoring metric measured each term's direct / indirect link's edge weights in the taxonomy (indicated by their corresponding values in the distance matrix), and aggregated them together in a weighted fashion to produce the final score. In this thesis, all of the following scoring schemes were used to evaluate taxonomies:

Average
This scheme calculates the score by taking the mean of all the direct edge weights in the taxonomy.

Momentum
This scheme calculates the score by taking the sum of the means of each term node's outgoing edges normalized by its incoming edge. A term node's incoming edge is the edge coming from its parent and its outgoing edges are the edges leading to its direct descendants.

Mean to Root
This scheme calculates the score by taking the sum of the means of each term node's edge weights to all its ancestors.

Mean to Grandparent
This scheme calculates the score by taking the sum of the means of each term node's edge weights up to two levels above (to its grandparent) node.

Linear
This scheme calculates the score by taking the sum of each term node's normalized linearly weighted distance to all its ancestors.

Exponential
This scheme calculates the score by taking the sum of each term node's normalized exponentially weighted distance to all its ancestors.

Figure 23 shows an example of scoring a taxonomy using the different scoring metrics.

Average: $\dfrac{(0.1 + 0.2 + 0.5 + 0.3 + 0.4)}{5} = \mathbf{0.3}$

Momentum: $(A_M + B_M + C_M + D_M + E_M + F_M)$
$= \dfrac{(0.1+0.2)}{2} + (\dfrac{(0.5+0.1)}{2} + \dfrac{(0.3+0.1)}{2}) + \dfrac{(0.4+ 0.2)}{2} + 0 + 0 + 0 = \mathbf{0.7}$

Mean to Root: $(A_{MTR} + B_{MTR} + C_{MTR} + D_{MTR} + E_{MTR} + F_{MTR})$
$= 0 + 0.1 + 0.2 + \dfrac{(0.5 + 0.1)}{2} + \dfrac{(0.3 + 0.1)}{2} + \dfrac{(0.4 + 0.2)}{2} = \mathbf{1.1}$

Mean to Grandparent: $(A_{MTG} + B_{MTG} + C_{MTG} + D_{MTG} + E_{MTG} + F_{MTG})$
$= 0 + 0.1 + 0.2 + \dfrac{(0.5 + 0.1)}{2} + \dfrac{(0.3 + 0.1)}{2} + \dfrac{(0.4 + 0.2)}{2} = \mathbf{1.1}$

Linear: $(A_L + B_L + C_L + D_L + E_L + F_L)$
$= 0 + 0.1 * 2/3 + 0.2 * 1/3 + (0.5 * 2/3 + 0.1 * 1/3) + (0.3 * 2/3 + 0.1 * 1/3) + (0.4 * 2/3 + 0.2 * 1/3)$
$= \mathbf{1.067}$

Exponential (with exponent 0.5) : $(A_E + B_E + C_E + D_E + E_E + F_E)$
$= 0 + 0.1 * 2/3 + 0.2 * 1/3 + (0.5 * 2/3 + 0.1 * 1/3) + (0.3 * 2/3 + 0.1 * 1/3) + (0.4 * 2/3 + 0.2 * 1/3)$
$= \mathbf{1.067}$
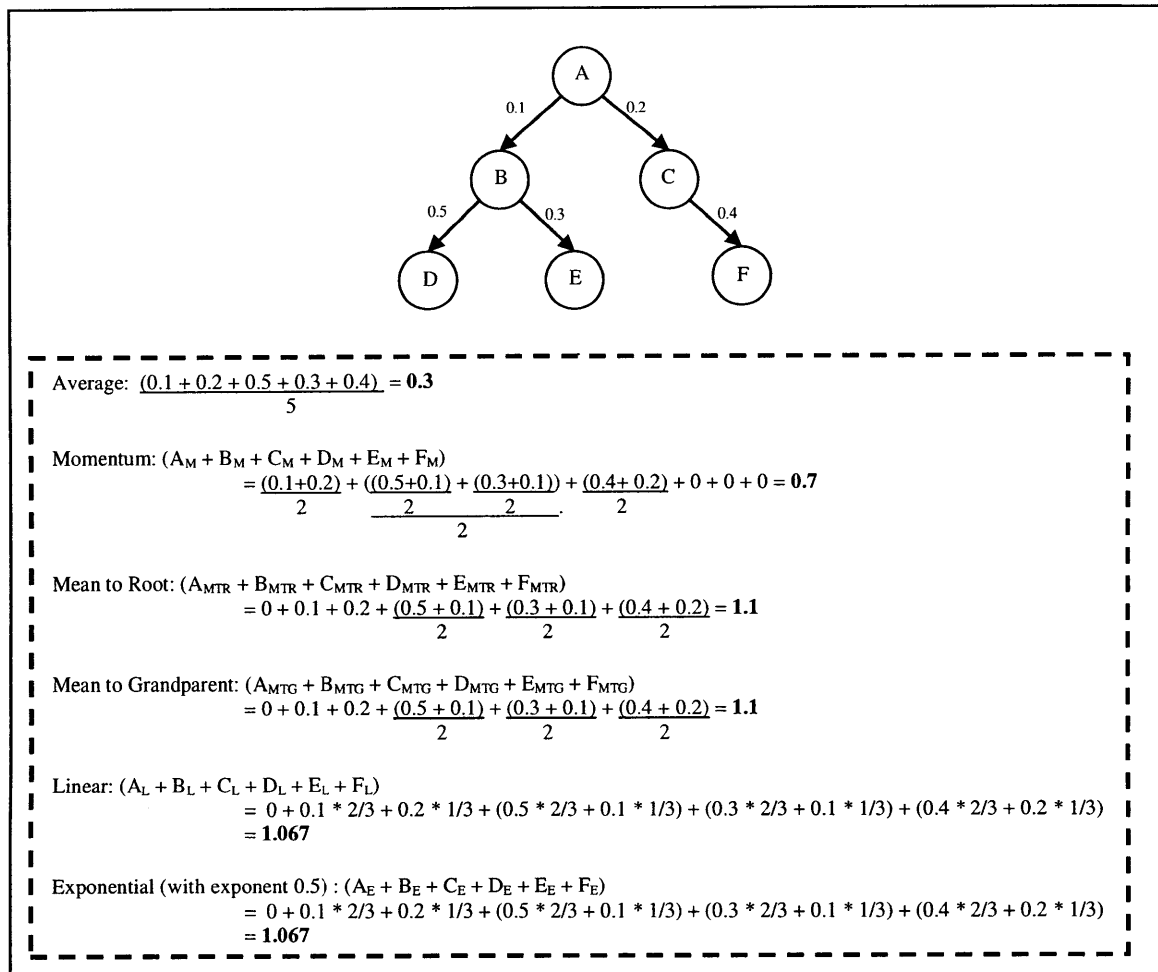
Figure 23: Example of Using Scoring Metrics to Score a Taxonomy

4.5 Synthetic Data Generation

The final test after checking each algorithm's internal consistency and evaluating each output's conformity to the distance matrix was to create a set of synthetic data, simulating a typical database one would realistically expect to gather from an online publication databases. The goal of this test is to see how well the taxonomy generation algorithms can reproduce underlying taxonomies.

This test is by no means easy to construct as there is no established way of generating synthetic data for taxonomy generation. However, there is a large body of literature relating to topic distribution / concept generation which we can adapt for the purpose of taxonomy generation. Specifically, our research group has looked closely into three classes of topic distribution / concept generation algorithms: Latent Sematic Analysis (LSA), Probabilistic Latent Semantic Analysis (PLSA), and Latent Dirichlet Allocation (LDA), all very mathematically rigorous. All three are statistical methods that analyze relationships between a set of documents and produce a set of topics / concepts (used interchangeably) related to the documents. LSA uses a term-document mathematical matrix that describes the occurrences of terms in documents and decomposes the matrix based on its singular value decomposition, in the end producing a set of vectors where each vector is a weighted set of terms corresponding to a concept. PLSA evolved from LSA and achieves a similar goal but in a different way. Whereas LSA's computations were grounded in linear algebra, PLSA's analysis is based on a latent class model, a statistical model that relates a set of variables to a set of latent variables. Finally, LDA is is similar to PLSA except that in LDA the topic distribution is assumed to have a Dirichlet Prior, which is a probability distribution that expresses the uncertainty about the topic distribution before the documents are taken into account. The basic principle behind these three analysis methods is that from a set of documents, a set of topics / concepts is generated, where each topic / concept is not composed of just one term, but rather is a weighted sum of terms. Applying this to suit our needs in synthetic data creation, we made it such that each topic / concept is represented by a single term, and each term has its own weighted distribution representing the degree to which all the other terms in the taxonomy affect it. In other words, whereas in LSA / PLSA / LDA each of the concepts are distinct from the terms they contain, for our purposes each term is also a concept.

To generate synthetic data, a predetermined, random taxonomy is first generated using a fixed set of terms. Each term in the taxonomy is then assigned its own probability distribution, which is a set of probabilities for each of the terms in the taxonomy to occur in a document whose central term is that term. As a hypothetical example, in a "renewable energy" related taxonomy, a document whose central term is "solar photovoltaics" will most likely have a probability distribution where "solar photovoltaics" has the highest chance of occurring in the document, while related terms like "solar", "renewable energy" and "solar energy" should also have some significant, nontrivial probability of occurring in the document.

For our purposes, we generated synthetic data with a fixed set of terms where each term has a probability distribution set to one where each term gets the highest probability of occurring in a document pertaining to itself, its ancestor terms in the taxonomy also get a significant non-

zero probability of occurring, and the rest of the nodes get a small non-zero probability of occurring.

Assigning a distribution to each term is a two phase process. First, each term is given an initial distribution where it is assigned a high probability p and the (1-p) probability is split among the rest of the terms in the distribution. Second, the individual distributions of the terms are aggregated by adding each term's distribution to the distribution of its parent and normalizing everything to a 0-1 scale. Figure 24 illustrates this process.
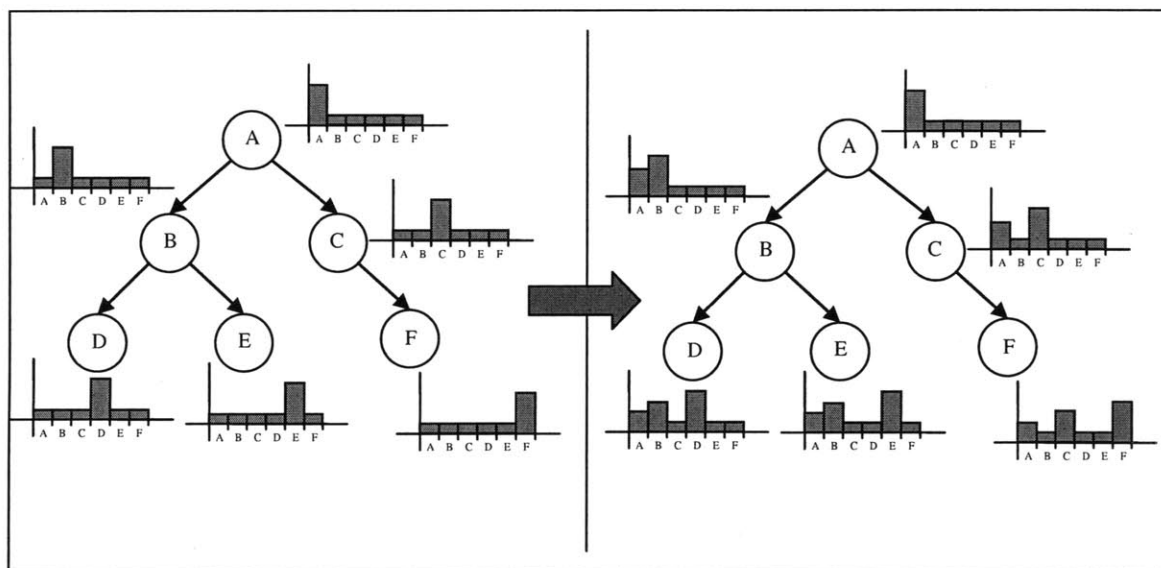
Figure 24: Assigning probability distributions for each of the terms in a taxonomy.

Next to each of the terms in the taxonomies shown above are the probability distributions associated to each term. The figure on the left shows the initial distribution assigned to each term, where all terms have their own term in their distribution having a significant amount of weight (indicated by a taller bar in the bar graph) and the other terms in the distribution having a much smaller amount. The figure on the right shows the final distribution assigned to each term, which takes into account the structure of the taxonomy. Notice how in each of the distributions aside from the root term, multiple terms have non-trivial weightings.

After each term's distribution is finalized, a set of documents are generated for each term based on the probability distributions for each term. For instance, in Figure 24 (b), a document generated for E will have a large probability that terms E, B, and A will be included as terms in the document. Since terms C, D and F only have a small probability within term E's distribution, the chance of them being included in a document relating to term E is slim. For a document generated relating to E, terms C, D and F are the *noise terms*. The probability of these terms being included can be increased by increasing the overall *noise* within the system. We define noise as the total probability associated to the non-related terms. For instance, if the noise was 0, then only terms E, B and A will ever occur in a document relating to term E, however if the noise was 1, then each of terms A, B, C, D, E and F will have an equal probability of occurring in a document relating to term E.

An equal amount of documents relating to each term are generated, and the collection of documents generated is used as the backend data set for the taxonomy generation algorithms. Essentially, this collection of documents is a simulated version of the expected collection from a real publication database.

The beauty of this process is that there is a predetermined underlying taxonomy for the set of documents generated, which is directly comparable to the taxonomies generated using the taxonomy generation algorithms developed in this thesis. Figure 25 illustrates a simple example of generating synthetic data.
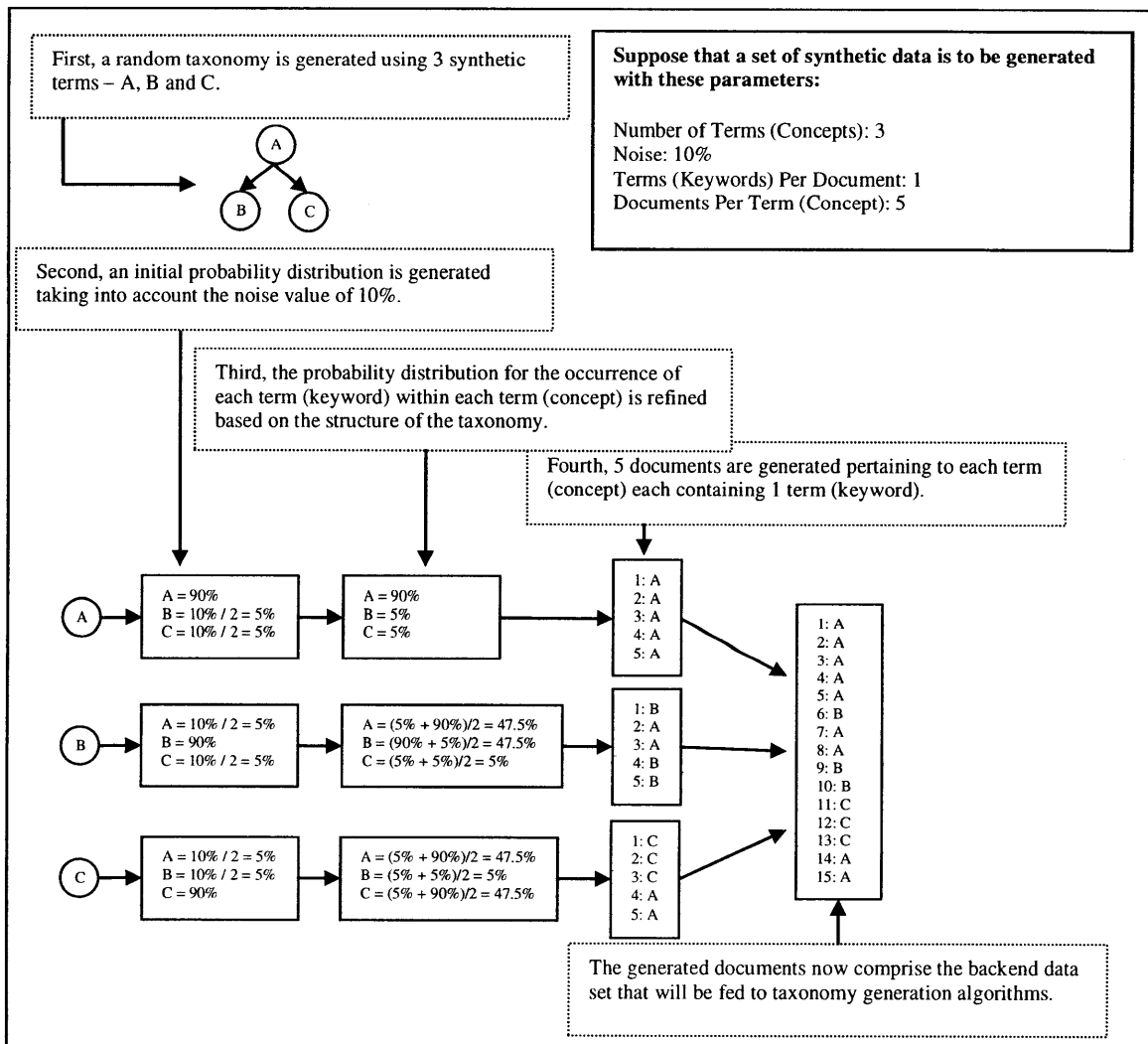
Figure 25: Synthetic Data Generation Process Example

This section explained the methodology used to analyze the taxonomy generation algorithms we've developed. The next section will discuss the results of the tests we conducted and give recommendations regarding which taxonomy generation algorithm(s) work best.

CHAPTER 5: Results

5.1 Introduction

In the previous chapter, the methodology for evaluating taxonomy generation algorithms was discussed. This chapter will present the results of those tests as well as highlight some interesting observations discovered in the process. At the end of this chapter, recommendations will be given regarding the best taxonomy generation algorithms among the options we have.

All the tests were run using a backend data set collected from Scopus using the following terms as the seed terms, after which duplicate articles were then removed from the database. For the rest of this thesis, we refer to this collection as the Scopus "renewable energy" data set:

1. "renewable energy"
2. "biodesel"
3. "biofuel"
4. "photovoltaic"
5. "solar cell"
6. "distributed generation"
7. "dispersed generation"
8. "distributed resources"

9. "embedded generation"
10. "decentralized generation"
11. "decentralized energy"
12. "distributed energy"
13. "on-site generation"
14. "geothermal"
15. "wind power"
16. "wind energy"

The data set collected contains 209,080 terms with 2,326 terms occurring more than 100 times among the entries, and 201 terms occurring more than 1,000 times. The top terms in the data set were gathered from among the INDEX_KEYWORDS in the Scopus data set for "renewable energy" as discussed in Chapter 3.2. The top 2,326 terms containing more than 100 occurrences are listed in Appendix A, however the first few are listed in Table 1 to give the reader a basic idea of the terms used in these tests:

Table 1: List of terms in Scopus "renewable energy" data set that have more than 2,500 occurrences in the data set.

Term Name	Number of Occurrences	Term Name	Number of Occurrences
solar-cell	23,268 occurrences	photovoltaic cell	11,865 occurrences
wind powers	15,776 occurrences	solar energy	8,847 occurrences
renewable energy resources	15,175 occurrences	biomass.	8,613 occurrences
energy policies	14,989 occurrences	mathematic models	7,173 occurrences
fuel cells	13,435 occurrences	computer simulator	6,860 occurrences

electrical power generations	6,216 occurrences	energy conversion	3,542 occurrences
geothermal energy.	5,958 occurrences	environmental impacts	3,500 occurrences
photovoltaic effect	5,957 occurrences	costs	3,425 occurrences
wind turbines	5,817 occurrences	silicon solar cells	3,324 occurrences
energy efficiencies	5,424 occurrences	electric utility	3,147 occurrences
electric generations	5,397 occurrences	electrode	3,115 occurrences
electrochemistry	5,338 occurrences	renewable resources	3,078 occurrences
thin film	5,230 occurrences	electrolyte	3,040 occurrences
electrical power system	4,814 occurrences	electrical potential	3,040 occurrences
hydrogen	4,767 occurrences	fossil fuels	3,036 occurrences
energy resources	4,755 occurrences	sustainable development	2,843 occurrences
energy utilities	4,549 occurrences	solid-oxide fuel cell	2,842 occurrences
article	4,403 occurrences	oxidizers	2,837 occurrences
power generation	4,339 occurrences	electric batteries	2,800 occurrences
energy conservation	3,988 occurrences	energy management	2,756 occurrences
optimizing	3,985 occurrences	methanol	2,717 occurrences
biofuels	3,925 occurrences	economizers	2,589 occurrences
eurasia	3,889 occurrences	catalysts	2,587 occurrences
electricity	3,676 occurrences	fuel	2,584 occurrences
carbon dioxide	3,626 occurrences	solar radiation	2,533 occurrences

As mentioned in the previous sections, by changing the parameters within each algorithms, a number of variants for each of the taxonomy generation algorithms can be created. Changing these parameters often leads to significantly different results and as such each variant needs to be treated as a separate algorithm. For each of the consistency tests run, all the taxonomy generation variants listed in

Table 2 were tested. For convenience, each algorithm was given an acronym by which it will be referred to for the rest of this section.

Table 2: List of Taxonomy Generation Variants

Algorithm Variant Acronym	Algorithm Type	Similarity Metric (used to create the distance matrix)	Centrality Metric Used (either to choose root or to decide term centrality at each iteration)
D-CB	DJP	Cosine	Betweenness
D-CC	DJP	Cosine	Closeness
D-SB	DJP	Symmetric NGD	Betweenness
D-SC	DJP	Symmetric NGD	Closeness
K-CB	Kruskals	Cosine	Betweenness
K-CC	Kruskals	Cosine	Closeness
K-SB	Kruskals	Symmetric NGD	Betweenness

K-SC	Kruskals	Symmetric NGD	Closeness
E-AB	Edmonds	Asymmetric NGD	Betweenness
E-AC	Edmonds	Asymmetric NGD	Closeness
H-AB	Heymann (with no rerunning of centrality metric, and no discarding of terms)	Asymmetric NGD	Betweenness
H-AC	Heymann (with no rerunning of centrality metric, and no discarding of terms)	Asymmetric NGD	Closeness
H-CB	Heymann (with no rerunning of centrality metric, and no discarding of terms)	Cosine	Betweenness
H-CC	Heymann (with no rerunning of centrality metric, and no discarding of terms)	Cosine	Closeness
H-SB	Heymann (with no rerunning of centrality metric, and no discarding of terms)	Symmetric NGD	Betweenness
H-SC	Heymann (with no rerunning of centrality metric, and no discarding of terms)	Symmetric NGD	Closeness

In addition, note that, as mentioned in Chapter 3, although the Genetic Algorithm is presented, it was decided not to include any test results for the algorithm since it is non-deterministic, which meant that every run of the algorithm generated results that varied significantly. While some preliminary testing was conducted on this algorithm, it was decided not to include these results since the inherent randomness of the Genetic algorithm makes the results unreproducible even if run under the same parameters.

We also decided not to show results of tests for the Heymann algorithm where the node centralities were recalculated after each iteration. This is because initial testing indicated that this variation to the algorithm did not seem to have much effect on the final taxonomy generated. In addition, allowing this modification resulted in a huge increase in the computational cost of generating the taxonomies. We concluded that the similarity between variant outcomes is due to the clustering of terms in the graph representation of the distance matrix. Recalculating the centrality metric at each iteration is most useful if the graph is highly segregated, containing several distinct central terms. In this thesis, the graphs generated are based on seed terms all

related to one central idea, "renewable energy". As such it is expected that the terms will be highly related to each other and hence very clustered. It is not necessary to recalculate the centralities for a highly clustered graph when using the Heymann algorithm.

Finally, we also decided not to show tests for the Heymann algorithm that allowed terms to be discarded. As explained in chapter 3.6.4, in our implementation of the Heymann algorithm, there is a threshold t_s that can be set to allow for the discarding of terms. The choice for the best threshold that generates the best taxonomy using the Heymann algorithm is completely dependent on the backend data set used. Within this thesis, we wanted to run tests to evaluate taxonomy generation algorithms that are applicable to any backend data set we may choose to use in the future, as such it was decided not to set a threshold thus not discarding any terms.

One important thing to keep in mind here is that all the algorithm variants tested in the succeeding sections run to completion in the same order of magnitude of time. As such, when analyzing the algorithms, not much concern is placed on the runtimes / speeds of the algorithms.

Outlined in this section are the results of the analysis run using the tests mentioned in the previous chapter. The first subsections list the results of the tests, then the last subsection analyzes the results generated.

5.2 Evaluating the Consistency of Taxonomy Generation Algorithms

As described in the previous chapter, the first set of tests were aimed at evaluating the consistency of the taxonomy generation algorithms. To do this, two sets of experiments were conducted to gauge the consistency, or robustness against noise, of the different taxonomy generation algorithms. The first set of experiments measured the consistency of the algorithms with slight perturbations in the backend, while in the second set of experiments, the backend database was fixed and perturbations were introduced to the collection of terms used to form the taxonomy.

5.2.1 Backend Data Set Consistency

For this test, the 153,537-entry bibliometric data set is randomly divided into five separate 100,000-entry subsets. The most popular terms from the entire Scopus "renewable energy" bibliometric data set were then taken and each of the taxonomy generation algorithms were run, keeping constant the term list and varying the backend data set between the five 100,000-entry sets. The percentage similarity of direct links between each of the taxonomies generated was then calculated between each of the 100,000-entry-backend data set taxonomies and the entire 153,537-entry-backend data set taxonomy. Table 3 summarizes the mean of the percentage similarities for each algorithm variant.

Table 3: Backend Data Set Consistency Test Results

Algorithm Variant Acronym	25 most frequently occurring terms used as term list	50 most frequently occurring terms used as term list	100 most frequently occurring terms used as term list	200 most frequently occurring terms used as term list	500 most frequently occurring terms used as term list	Mean of Percentage Similarities
D-CB	77.60%	98.00%	97.80%	95.80%	94.08%	92.66%
D-CC	94.40%	97.60%	97.60%	95.80%	94.08%	**95.90%**
D-SB	92.80%	96.80%	94.40%	91.70%	91.00%	93.34%
D-SC	93.60%	96.80%	94.40%	91.10%	91.00%	93.38%
K-CB	77.60%	98.00%	97.80%	95.80%	93.88%	92.62%
K-CC	94.40%	97.60%	97.60%	95.80%	93.88%	**95.86%**
K-SB	4.00%	2.00%	1.00%	0.50%	0.20%	1.54%
K-SC	4.00%	2.00%	1.00%	0.30%	0.20%	1.50%
E-AB	90.40%	93.20%	90.60%	88.90%	84.64%	89.55%
E-AC	93.60%	93.20%	90.60%	88.90%	84.52%	90.16%
H-AB	88.00%	92.80%	96.20%	96.30%	97.48%	94.16%
H-AC	95.20%	**98.40%**	98.00%	**97.50%**	**97.68%**	**97.36%**
H-CB	34.40%	36.80%	29.00%	33.30%	29.92%	32.68%
H-CC	96.00%	97.60%	**98.60%**	95.90%	94.96%	**96.61%**
H-SB	78.40%	73.20%	78.60%	83.90%	82.96%	79.41%
H-SC	**96.80%**	96.40%	94.60%	93.80%	91.56%	94.63%

Highlighted in the table above are the top performers for each test run. Based on these results, the best performing algorithm variants (over 95% similarity on average) are:

1. Heymann algorithm, asymmetric NGD metric, closeness centrality (H-AC)

2. DJP algorithm, cosine similarity, closeness centrality for root selection (D-CC)

3. Kruskals algorithm, cosine similarity, closeness centrality for root selection (K-CC)

4. Heymann algorithm, cosine similarity, closeness centrality (H-CC)

Other notable observations are:

1. The use of Kruskals algorithm with symmetric NGD similarity is not a consistent algorithm at all. It was barely able to create a single consistent link between the taxonomies generated using the 100,000-entry-backends and the 153,537-entry-backend.

2. The tests for the Heymann algorithm all show that the use of closeness centrality is a much more consistent metric than using betweenness centrality. Note that the differences between closeness and betweenness centrality are only evident when examining the

results of the Heymann algorithm tests because Heymann is the only taxonomy generation algorithm that uses the centrality measures for more than just picking the root node.

5.2.2 Term Consistency

For this test, the backend was kept constant, and consisted of the entire 153,537-entry Scopus "renewable energy" bibliometric data set. However, the term lists were varied by taking the most popular terms in the data set and inserting "noise" terms, which are terms selected randomly from the rest of the terms in the data set. We chose to insert an equal number of noise terms to the terms already in the taxonomy. For instance, if a taxonomy was created using the 25 most frequently occurring terms, 25 noise terms were inserted into the taxonomy, then each taxonomy generation algorithm was run using those 50 total terms, and percentage of the number of links consistent in the 25-term noise-free and 50-term noisy taxonomies outputted by each taxonomy generation algorithm was calculated. Comparing these two taxonomies required simplifying the larger 50-term taxonomy using the method mentioned in the previous chapter. This test was repeated three times and the mean of the three percentage link similarities for each algorithm was taken. The results are summarized in Table 4.

Table 4: Term Consistency Test Results

Algorithm Variant Acronym	25 most frequently occurring terms, with 25 more noise terms	50 most frequently occurring terms, with 50 more noise terms	100 most frequently occurring terms, with 100 more noise terms	250 most frequently occurring terms, with 250 more noise terms	Mean of Percentage Similarities
D-CB	76.92%	**97.39%**	87.79%	86.06%	87.04%
D-CC	92.31%	**97.39%**	87.79%	86.06%	**90.88%**
D-SB	**94.87%**	88.24%	82.51%	77.69%	85.83%
D-SC	87.18%	94.12%	84.82%	81.01%	86.78%
K-CB	76.92%	**97.39%**	87.79%	86.06%	87.04%
K-CC	92.31%	**97.39%**	87.79%	86.06%	**90.88%**
K-SB	7.69%	31.37%	0.66%	0.00%	9.93%
K-SC	0.00%	35.29%	1.32%	0.27%	9.22%
E-AB	75.64%	95.42%	83.83%	82.20%	84.27%
E-AC	80.77%	95.42%	79.87%	80.88%	84.23%
H-AB	64.10%	86.93%	86.14%	86.59%	80.94%
H-AC	83.33%	94.77%	83.83%	87.38%	87.33%
H-CB	23.08%	37.25%	27.39%	33.33%	30.26%
H-CC	91.03%	94.12%	**89.44%**	**88.58%**	**90.79%**
H-SB	50.00%	54.90%	60.07%	55.78%	55.19%
H-SC	75.64%	72.55%	78.22%	76.49%	75.73%

Highlighted in the table above are the top performers for each test run. The best performing algorithm variants (over 90% similarity) based on our tests are:

1. DJP algorithm, cosine similarity, closeness centrality for selecting the root (D-CC)

2. Kruskals algorithm, cosine similarity, closeness centrality for selecting the root (K-CC)

3. Heymann algorithm, cosine similarity, closeness centrality (H-CC)

Other notable observations from this test are:

5. The use of Kruskals algorithm with symmetric NGD similarity is not a consistent algorithm at all. It was barely able to create a single consistent link when noise terms were inserted.

6. The tests for the Heymann algorithm all show that the use of closeness centrality is a much more consistent metric than using betweenness centrality.

5.2.3 Consistency Test Summary

The consistency tests were run both by varying the backend data set and term lists to test for taxonomy generation algorithm robustness versus noise. Table 5 repackages the information from Table 3 and Table 4 for easier viewing.

Table 5: Consistency Test Summary

Algorithm Variant Acronym	Mean of Percentage Similarities for Backend Data Set Consistency Test	Mean of Percentage Similarities for Term List Consistency Test
D-CB	92.66%	87.04%
D-CC	**95.90%**	**90.88%**
D-SB	93.34%	85.83%
D-SC	93.38%	86.78%
K-CB	92.62%	87.04%
K-CC	**95.86%**	**90.88%**
K-SB	1.54%	9.93%
K-SC	1.50%	9.22%
E-AB	89.55%	84.27%
E-AC	90.16%	84.23%
H-AB	94.16%	80.94%
H-AC	**97.36%**	87.33%
H-CB	32.68%	30.26%
H-CC	**96.61%**	**90.79%**

H-SB	79.41%	55.19%
H-SC	94.63%	75.73%

Based on the results shown in Table 5 above, the following is clear:

1. The use of Kruskals algorithm with symmetric NGD similarity is not a consistent algorithm in any way.

2. Closeness centrality seems to be a much better similarity metric compared to betweenness centrality.

3. The most consistent algorithms variants are D-CC, K-CC and H-CC, all of which use cosine similarity and closeness centrality to generate taxonomies.

5.3 Evaluating Individual Taxonomies

Several tests were run which tested each of the taxonomy generation algorithms' outputs individually by taking their outputs and scoring them using the different scoring metrics described in the previous chapter. To recap, the scoring metrics used were (for more information about each of the metrics mentioned above, see the previous chapter of this thesis):

1. Average

2. Momentum

3. Mean to Root

4. Mean to Grandparent

5. Linear

6. Exponential (0.5)

7. Exponential (0.75)

Note that the scoring algorithms measure each taxonomy's conformity to its distance matrix and as such are only useful when comparing taxonomies generated using the same similarity metric since only one similarity metric characterizes a distance matrix. This means that using a given scoring metric, it is impossible to compare all the taxonomy generation algorithms to each other, however it is possible to compare all the taxonomy generation algorithms that used the cosine similarity metric, symmetric NGD similarity metric, or asymmetric NGD similarity metric to each other.

The top 100, 250 and 500 frequently occurring terms in the Scopus "renewable energy" data set were used in conjunction with the entire bibliometric data set . The results are presented in the following subsections.

5.3.1 Using the top 100 terms

The results summarized in Table 6 are from tests run using the cosine similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Table 6: Different Scoring Metrics used on Cosine Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-CB	**0.341**	27.632	20.835	27.632	24.066	26.961	23.533
D-CC	**0.341**	27.912	22.701	27.912	25.375	27.884	24.994
K-CB	**0.341**	27.632	20.835	27.632	24.066	26.961	23.533
K-CC	**0.341**	27.912	22.701	27.912	25.375	27.884	24.994
H-CB	0.285	23.340	21.232	23.340	23.391	24.112	22.487
H-CC	0.337	**28.272**	**24.805**	**28.272**	**27.305**	**28.587**	**26.460**

The results summarized in Table 7 are from tests run using the symmetric NGD similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Table 7: Different Scoring Metrics used on Symmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-SB	**0.110**	13.126	20.918	13.126	18.366	14.085	16.925
D-SC	**0.110**	**13.045**	20.447	**13.045**	18.380	**13.922**	16.665
K-SB	0.323	27.662	27.662	27.662	29.214	29.214	28.327
K-SC	0.323	26.918	26.918	26.918	28.719	28.719	27.690
H-SB	0.119	14.451	17.806	14.451	15.998	14.898	16.471
H-SC	0.114	13.566	**17.307**	13.566	**15.680**	14.183	**15.857**

Finally, the results summarized in Table 8 are from tests run using the asymmetric similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Table 8: Different Scoring Metrics used on Asymmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
E-AB	**0.028**	**3.651**	**3.758**	**3.651**	**3.431**	**3.410**	**3.608**
E-AC	**0.028**	**3.651**	**3.758**	**3.651**	**3.431**	**3.410**	**3.608**
H-AB	0.028	3.705	3.791	3.705	3.470	3.449	3.644
H-AC	**0.028**	**3.651**	**3.758**	**3.651**	**3.431**	**3.410**	**3.608**

5.3.2 Using the top 250 terms

The results summarized in Table 9 are from tests run using the cosine similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Table 9: Different Scoring Metrics used on Cosine Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-CB	**0.333**	67.507	52.780	67.507	60.950	66.572	58.883
D-CC	**0.333**	**67.765**	53.575	**67.765**	61.082	**66.864**	59.388
K-CB	**0.333**	67.507	52.780	67.507	60.950	66.572	58.883
K-CC	**0.333**	**67.765**	53.575	**67.765**	61.082	**66.864**	59.388
H-CB	0.286	57.262	47.932	57.262	54.345	57.773	52.211
H-CC	0.327	67.260	**54.777**	67.260	**62.308**	66.696	**60.048**

The results summarized in Table 10 are from tests run using the symmetric NGD similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Table 10: Different Scoring Metrics used on Symmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-SB	**0.112**	34.048	55.832	34.048	48.896	36.804	45.164
D-SC	**0.112**	**33.882**	52.860	**33.882**	47.140	**36.515**	44.071
K-SB	0.335	73.046	73.046	73.046	76.584	76.584	74.562
K-SC	0.335	72.651	72.651	72.651	76.321	76.321	74.224
H-SB	0.122	38.392	51.256	38.392	45.995	40.584	46.358
H-SC	0.117	35.266	**49.411**	35.266	**44.013**	37.741	**43.784**

Finally, the results summarized in Table 11 are from tests run using the asymmetric NGD similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Table 11: Different Scoring Metrics used on Asymmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
E-AB	**0.013**	**7.233**	**8.579**	**7.233**	**6.917**	**6.578**	**7.719**
E-AC	**0.013**	**7.233**	**8.579**	**7.233**	**6.917**	**6.578**	**7.719**
H-AB	0.013	7.820	8.680	7.820	7.067	6.758	7.861
H-AC	**0.013**	7.733	8.654	7.733	7.039	6.722	7.830

5.3.3 Using the top 500 terms

The results summarized in Table 12 are from tests run using the cosine similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Table 12: Different Scoring Metrics used on Cosine Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-CB	**0.316**	124.705	89.254	124.705	105.961	121.457	103.552
D-CC	**0.316**	**124.998**	90.791	**124.998**	106.101	122.087	104.567
K-CB	**0.316**	124.705	89.254	124.705	105.961	121.457	103.552
K-CC	**0.316**	**124.998**	90.791	**124.998**	106.101	122.087	104.567

H-CB	0.260	100.394	76.155	100.394	88.615	100.057	86.514
H-CC	0.309	124.609	**95.556**	124.609	**111.186**	**122.363**	**107.433**

The results summarized in Table 13 are from tests run using the symmetric NGD similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Table 13: Different Scoring Metrics used on Symmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-SB	**0.116**	71.257	138.022	71.257	118.328	78.324	100.547
D-SC	**0.116**	**71.123**	115.215	**71.123**	103.762	**77.345**	94.929
K-SB	0.361	170.407	170.407	170.407	173.822	173.822	171.871
K-SC	0.361	158.177	158.177	158.177	165.669	165.669	161.388
H-SB	0.126	79.727	119.995	79.727	104.705	85.882	102.732
H-SC	0.121	74.192	**105.994**	74.192	**93.791**	79.285	**92.860**

Finally, the results summarized in Table 14 are from tests run using the asymmetric NGD similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Table 14: Different Scoring Metrics used on Asymmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
E-AB	0.015	15.659	**18.957**	15.659	15.374	14.441	17.005
E-AC	0.015	14.375	19.668	14.375	16.034	14.428	17.373
H-AB	0.015	15.737	19.013	15.737	15.444	14.516	17.076
H-AC	**0.015**	**14.165**	19.108	**14.165**	**15.275**	**14.059**	**16.869**

5.3.4 Evaluating Individual Taxonomies Analysis

The consistently top-scoring algorithms among the 100, 250 and 500 term list tests are summarized in Table 15. The shaded cells represent the consistently top-scoring algorithm variants for each of the scoring metrics.

Table 15: Consistently Top Scoring Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grand-parent	Linear	Exponential (0.5)	Exponential (0.75)
D-CB	▓						
D-CC	▓						
D-SB	▓						
D-SC		▓		▓		▓	
K-CB	▓						
K-CC	▓						
K-SB							
K-SC							
E-AB			▓				
E-AC							
H-AB							
H-AC	▓						
H-CB							
H-CC			▓		▓		▓
H-SB							
H-SC			▓		▓		▓

Based on the data in the table above, the algorithm variant that performed the best is:

1. DJP algorithm, symmetric NGD similarity, cosine centrality for root selection (D-SC)

Other notable top performing algorithm variants were:

1. Heymann algorithm, symmetric NGD similarity, closeness centrality (H-SC)

2. Heymann algorithm, cosine similarity, closeness centrality (H-CC)

Aside from the results summarized in Table 15, it is also clear through the Heymann algorithm tests that closeness centrality seems to be a better centrality metric to use compared to betweenness centrality. This is consistent with our observations in the previous consistency tests.

5.4 Synthetic Data Generation

Finally, as explained in Chapter 4.5, synthetic data sets were generated to allow the taxonomy generation algorithms to be tested on data with known characteristics. This allowed two different sets of tests to be conducted. Firstly, the optimal range of sizes for bibliometric data sets from which taxonomies could be accurately inferred was estimated. The second set of tests was designed to study the performance of taxonomy generation algorithms when faced with different noise levels.

5.4.1 Estimating the Optimal Bibliometric Data Set Size

The first challenge was to estimate the optimal size for a synthetically produced taxonomy. This was done by creating underlying taxonomies of different sizes, then generating a varying amount of documents for each term in the taxonomy, which were then fed as input to each of the taxonomy generation algorithms. The outputs of each taxonomy generation algorithm were then compared to the valid, predetermined underlying taxonomies.

For these tests, a noise level of 0.2 was assumed within the documents. As mentioned in the previous chapter, this is defined as the probabilities of the "off-topic" terms relative to the probability of the relevant term. 0.2 was our subjective reasonable estimate for the noise level to be expected from a real publication database.

The two tables in the following pages summarize the results – note that all experiments were repeated three times to take into account the variance of generating random initial synthetic taxonomies, and the average scores reported. Table 16 lists the percentage similarity of the synthetically produced underlying taxonomies to the taxonomies generated using algorithms that use betweenness centrality while Table 17 lists the ones for closeness centrality. Highlighted are the best performing data set sizes for each taxonomy generation algorithm.

Table 16: Accuracy of Taxonomy Generation Algorithms Using Betweenness Centrality's Outputs for Replicating Underlying Synthetically Generated Taxonomies

Total Entries in Data Set	Number of Terms	D-CB	D-SB	K-CB	K-SB	E-AB	H-AB	H-CB	H-SB
1000	20	78.33%	78.33%	78.33%	21.67%	45.00%	33.33%	**5.00%**	35.00%
2000	20	83.33%	86.67%	83.33%	21.67%	36.67%	26.67%	**5.00%**	23.33%
2,500	50	92.67%	89.33%	92.67%	10.00%	76.00%	60.67%	2.00%	52.00%
5,000	50	94.67%	95.33%	94.67%	10.00%	64.67%	48.00%	2.00%	51.33%
5,000	100	95.00%	91.33%	95.00%	8.33%	77.00%	62.00%	1.00%	50.67%
10,000	100	94.67%	96.33%	94.67%	5.00%	**89.67%**	**73.67%**	1.00%	**58.33%**
20,000	20	81.67%	91.67%	81.67%	20.00%	35.00%	26.67%	**5.00%**	8.33%
50,000	50	91.33%	93.33%	91.33%	13.33%	74.67%	47.33%	2.00%	33.33%
100,000	20	80.00%	83.33%	80.00%	23.33%	48.33%	28.33%	**5.00%**	10.00%
100,000	100	**95.67%**	**97.33%**	**95.67%**	8.00%	76.67%	51.00%	1.00%	42.67%
200,000	20	85.00%	86.67%	85.00%	**30.00%**	40.00%	33.33%	**5.00%**	6.67%
250,000	50	87.33%	92.67%	87.33%	8.00%	84.00%	57.33%	2.00%	38.67%
500,000	50	92.67%	96.00%	92.67%	9.33%	74.67%	52.00%	2.00%	41.33%
500,000	100	94.33%	96.67%	94.33%	6.67%	81.00%	55.33%	1.00%	41.67%
1,000,000	100	95.00%	97.00%	95.00%	4.67%	88.00%	59.67%	1.00%	38.00%

Table 17: Accuracy of Taxonomy Generation Algorithms Using Closeness Centrality's Outputs for Replicating Underlying Synthetically Generated Taxonomies

Total Entries in Data Set	Number of Terms	D-CC	D-SC	K-CC	K-SC	E-AC	H-AC	H-CC	H-SC
1000	20	86.67%	90.00%	86.67%	18.33%	45.00%	36.67%	73.33%	68.33%
2000	20	93.33%	95.00%	93.33%	16.67%	36.67%	35.00%	80.00%	76.67%
2,500	50	97.33%	95.33%	97.33%	8.67%	76.00%	70.67%	86.67%	72.67%
5,000	50	98.00%	96.67%	98.00%	8.67%	64.67%	66.00%	83.33%	77.33%
5,000	100	98.67%	94.67%	98.67%	7.33%	77.00%	73.67%	82.00%	61.00%
10,000	100	98.67%	**99.00%**	98.67%	4.00%	**89.67%**	88.00%	86.67%	67.00%
20,000	20	95.00%	95.00%	95.00%	16.67%	35.00%	35.00%	83.33%	**81.67%**
50,000	50	98.00%	98.00%	98.00%	11.33%	74.67%	71.33%	86.67%	73.33%
100,000	20	95.00%	95.00%	95.00%	18.33%	48.33%	46.67%	81.67%	80.00%
100,000	100	**99.00%**	**99.00%**	**99.00%**	7.33%	76.67%	75.33%	88.33%	49.00%
200,000	20	95.00%	95.00%	95.00%	**26.67%**	40.00%	36.67%	88.33%	**81.67%**
250,000	50	98.00%	98.00%	98.00%	6.00%	85.33%	82.67%	87.33%	80.67%
500,000	50	98.00%	98.00%	98.00%	8.00%	74.67%	75.33%	90.00%	72.67%
500,000	100	**99.00%**	**99.00%**	**99.00%**	5.67%	81.33%	82.33%	90.00%	44.33%
1,000,000	100	**99.00%**	**99.00%**	**99.00%**	3.67%	88.00%	**90.00%**	**90.67%**	52.33%

Based on the tables above, we note several key observations:

1. DJP and Kruskals algorithm variants have the general trend where the more terms in the taxonomy or the more entries in the bibliometric data set exist, the more accurate the replication of the underlying taxonomy is.

2. The algorithms that use cosine similarity perform much better than the other algorithm variants.

3. Using the closeness centrality metric produces much more accurate results than using the betweenness centrality. The disparity between the two is evident in the tests run using the Heymann algorithm.

The results of the tests that used closeness similarity are summarized in Table 18.

Table 18: Average of Closeness Centrality Algorithms Accuracy Results

Total Entries in Data Set	Number of Terms	Average of Percentage Similarities for all Taxonomy Generation Algorithms
1000	20	63.13%
2000	20	65.83%
2,500	50	75.58%
5,000	50	74.08%
5,000	100	74.13%
10,000	100	78.96%
20,000	20	67.08%
50,000	50	76.42%
100,000	20	70.00%
100,000	100	74.21%
200,000	20	69.79%
250,000	50	**79.50%**
500,000	50	76.83%
500,000	100	75.08%
1,000,000	100	72.71%

As shown in the table above, taxonomy generation algorithms on average are most accurate (best replicate the underlying taxonomy) when there are 50 terms and 250,000 bibliometric entries in the data set. Past this value the mean of the accuracy of the taxonomy generation algorithms decreases. As such, for the tests in the next section where we varied the noise, we considered the scenario where there were 250,000 total entries in the data set.

5.4.2 Measuring Algorithm Variant Consistency Using Synthetic Data

Using a predetermined, underlying taxonomy with size based on the findings in the previous section, the "noise" values within the data set were varied to calculate the robustness vs noise, or consistency of each taxonomy generation algorithm.

A data set was created consisting of 50 terms with 5,000 entries generated for each term, totaling to 250,000 entries in the synthetic data set. The test was run three times and the percentage similarity values were averaged. The results of this test are summarized in Table 19 below. Note that the percentage values represent the degree of similarity of the outputs of each taxonomy generation algorithm to the underlying taxonomy. Highlighted values represent the best performing algorithms for every noise value.

Table 19: Accuracy of Taxonomy Generation Algorithms for Replicating Underlying Synthetically Generated Taxonomies with 50 Terms with Varying Noise

Algorithm Variant Acronym	Noise = 0	Noise = 0.2	Noise = 0.5	Noise = 0.8	Noise = 1	Average	Std Dev (does not count case where noise = 1)
D-CB	94.00%	90.00%	84.67%	13.33%	2.00%	56.80%	38.30%
D-CC	**98.00%**	**98.00%**	90.67%	11.33%	2.00%	60.00%	42.25%
D-SB	**98.00%**	95.33%	92.00%	89.33%	2.00%	75.33%	3.79%
D-SC	**98.00%**	**98.00%**	**98.00%**	**96.67%**	2.00%	**78.53%**	**0.67%**
K-CB	94.00%	90.00%	84.67%	13.33%	**15.33%**	59.47%	38.30%
K-CC	**98.00%**	**98.00%**	90.67%	11.33%	**15.33%**	62.67%	42.25%
K-SB	8.67%	11.33%	10.67%	8.00%	**15.33%**	10.80%	1.59%
K-SC	8.67%	9.33%	8.67%	6.67%	**15.33%**	9.73%	1.15%
E-AB	96.67%	80.67%	21.33%	8.00%	**15.33%**	44.40%	43.56%
E-AC	96.67%	80.67%	21.33%	6.00%	**15.33%**	44.00%	44.24%
H-AB	80.67%	56.67%	2.00%	2.00%	2.00%	28.67%	39.72%
H-AC	96.67%	78.67%	19.33%	6.00%	2.00%	40.53%	44.26%
H-CB	2.00%	2.00%	2.00%	2.00%	2.00%	2.00%	**0.00%**
H-CC	90.67%	91.33%	86.00%	9.33%	2.00%	55.87%	40.07%
H-SB	64.00%	34.00%	2.00%	2.00%	2.00%	20.80%	29.77%
H-SC	**98.00%**	75.33%	86.00%	89.33%	2.00%	70.13%	9.37%

Based on the data in the table above, the best performing and most robust algorithm vs noise was:

1. DJP algorithm, symmetric NGD similarity, closeness centrality for choosing the root term (D-SC).

It consistently managed to replicate most of the links in the underlying taxonomy and had a low variance in its percentage accuracy as the noise values were varied.

5.5 Analysis of Results

The tests within this section rigorously tested each taxonomy generation algorithm variant. It was decided not to test the Genetic Algorithm (GA), since this algorithm produced outputs that were just too different from each other, hence it was not a worthwhile taxonomy generation algorithm to examine. The GA did use several taxonomy scoring metrics, which were used in the succeeding tests.

The first set of tests conducted measured each algorithm variant's consistency, or robustness vs noise. The consistency tests were further subdivided into backend data consistency tests and term list consistency tests. From the first set of tests, it was discovered that the most consistent algorithm variants were D-CC, K-CC and H-CC, all of which use cosine similarity and closeness centrality to generate taxonomies. The fact that these three variants were the most consistent seem to show that the cosine similarity metric and closeness centrality are effective algorithm parameters as well.

The second set of tests conducted evaluated individual taxonomies based on several scoring metrics that measured each taxonomy generation algorithm variant's conformity to its distance matrix. Each distance matrix is built using a particular similarity metric, so one downside of this test was that it was impossible to compare algorithms that used different similarity metrics to generate their distance matrices. Among the algorithm variants, the consistent top performer was D-SC, followed by H-CC and H-SC. Once again, closeness centrality was the metric all the efficient algorithm variants used to generate their taxonomies, however this time the symmetric NGD metric was used by the top performer to generate its distance matrix. Similar to cosine similarity, Symmetric NGD similarity is another similarity metric that produces an undirected graph. This seems to indicate that the most consistent and top-scoring algorithm variants use similarity metrics that are undirected.

Finally, synthetic data sets were generated based on known, randomly generated taxonomies and were used to measure the respective performances of each of the taxonomy generation algorithm variants in replicating the underlying taxonomy. The first set of synthetic data tests showed that the ideal data set size for which our algorithms can accurately produce valid taxonomies consistently is 250,000 entries. Then, using this data set size, the noise within the data was varied and each taxonomy generation algorithm variant's robustness vs noise was measured. From these tests the best performing algorithm variant was found to be D-SC.

Based on all the tests conducted, there is now convincing evidence that the best algorithm variants are H-CC and D-SC, since these were the two algorithm variants that performed well in multiple (two out of three) tests. As a final focus for analysis, we manually inspected each well-performing algorithm variant's outputs to determine which taxonomy generation algorithm works best. Specifically, we inspected the taxonomies generated by H-CC and D-SC using the entire

"renewable energy" Scopus data set as backend, and used the top 500 frequently occurring terms in the data set as term list.

The figures on the succeeding pages show the taxonomies generated by both the algorithm variants. Specifically Figure 26 shows the H-CC taxonomy and Figure 27 shows the D-SC taxonomy.

One main observation that is immediately clear upon inspection of the D-SC taxonomy is that it is very deep, going as far as 25 levels in. Note that the root of the taxonomy in the figure is on the lefthand-side and as such a deeper taxonomy would be a wider / broader figure. In contrast to the D-SC taxonomy, the H-CC taxonomy is not very deep, though it still goes 5-9 levels in.

Upon a more granular inspection of both taxonomies, it seems that the taxonomy generated using the H-CC algorithm makes a little more sense. Both taxonomies generated using H-CC and D-SC used the same term list, but the taxonomy generated using D-SC did not have any clear clustering of terms that represented the same idea, whereas the one generated using H-CC had clear term clusters, which are indicated in Figure 26. The lack of clustering in the D-SC taxonomy is also a by-product of its depth. Since it is very deep, it isn't very broad, hence each term only has on average 3 children, and hence it's harder to immediately notice term clusters.

Even though the H-CC taxonomy looks more sensible than the D-SC taxonomy, it is by no means perfect. For instance, there are clusters in the taxonomy that grouped seemingly unrelated terms together. An example of this was a cluster of terms where the parent node was "ph", a chemistry-related term referring to the acidity of a solution, however its children were "solar" related like "photovoltaic", "spectrum analysis" and "photoconduction" as well as chemistry-related terms like "solute".

On the other hand, the taxonomy generated using D-SC also had it advantages. Within the taxonomy, certain logical paths could be traced. For instance, starting from "power generation", we can trace the following path by going deeper in the taxonomy: "power generation" → "electric powers" → "power system" → "electrical power system" → "power transmission" → "electric power transmission" → "electric power transmission networks" → "electric network analysis". The location of the path in the taxonomy is seen in Figure 27.

As such, we have concluded that both the H-CC and the D-SC produced taxonomies that provide useful information in different ways. The choice of which taxonomy generation algorithm variant to use is dependent on the viewer's preferences. If a taxonomy is desired that separates distinct term concepts, then H-CC should be used. If on the other hand a taxonomy is desired where long paths can be traced between related terms, then D-SC should be used.

There were a few more observations in the results section that we now offer possible explanations for. Firstly, in all our tests, the Kruskal's algorithm taxonomy generation variants always performed terribly. We believe that this is most likely due to Kruskal's choosing of the

root term at the end of the algorithm. The top performing algorithms (H-CC and D-SC) choose the root terms at the beginning of the algorithm, and all the other links are added already assuming the position of the root term at the head of the taxonomy. The choice of the root term by Kruskal's only at the end of the algorithm allows for the inclusion of several links in the taxonomy that otherwise would not have been included if the root term was selected first. Hypothetically, if Kruskal's is modified such that it selects the root term first, it essentially turns into the DJP algorithm, which is the basis of the D-SC algorithm variant.

Secondly, in all our tests, the variants that use the undirected / symmetric similarity metric (cosine and symmetric NGD) outperform the directed / asymmetric similarity variants (asymmetric NGD). We believe that this is likely due to the additional flexibility provided by a symmetric similarity metric. A symmetric similarity metric allows for a pair of terms to be linked in two ways (ex: terms **X** and **Y** can be linked either **X→Y** or **X←Y**), whereas an asymmetric similarity metric only allows for one direction for linking. As such, if given the choice, a taxonomy generation algorithm that uses an asymmetric similarity metric could more likely link two terms that are unrelated than link two related terms that have the reverse parent-child relationship, whereas a taxonomy generation algorithm that uses a symmetric similarity metric would go ahead and link the related terms since it has no notion of directionality or the parent-child relationship.

Finally, in all our tests, the variants that use closeness centrality metric outperform the variants that use betweenness centrality. We believe that this is because betweenness centrality only takes into account the fraction of the number of shortest paths that pass through a node in a graph, whereas closeness centrality actually looks at the shortest path lengths and takes that value into account. As such, the comparison value used by closeness centrality is more granular than the one used by betweenness centrality, leading to a more accurate measure of centrality.
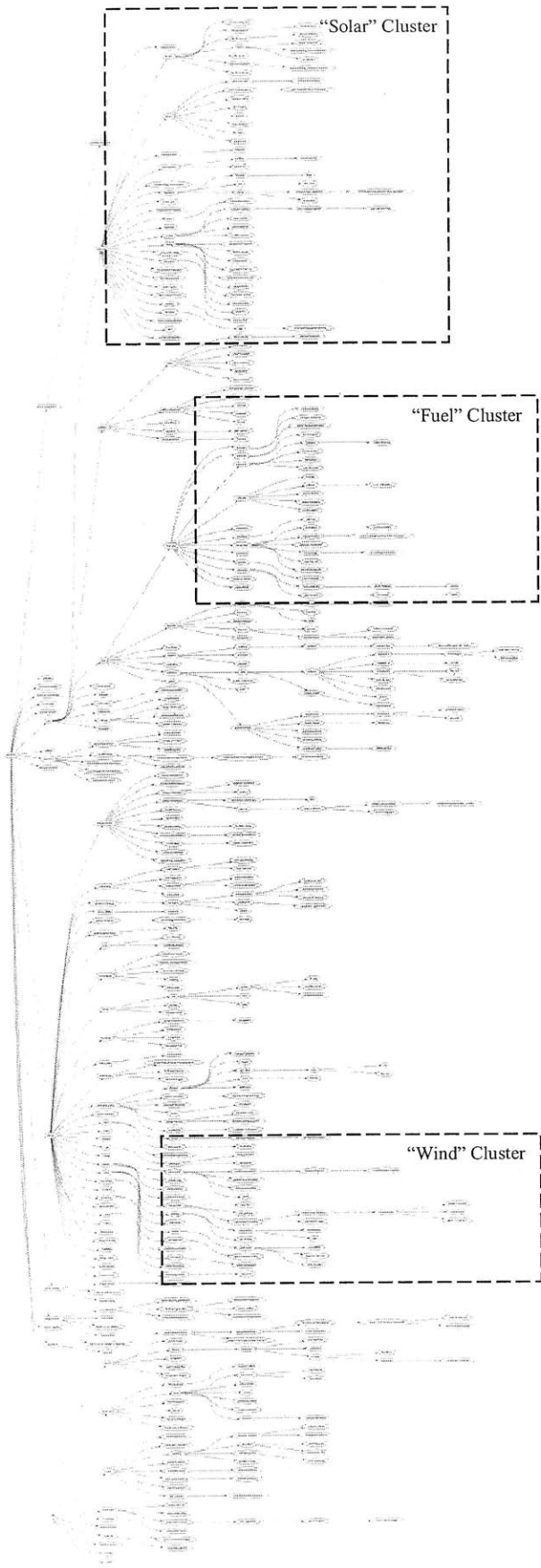
Figure 26: Visual Representation of HCC-Generated Taxonomy

"Power Generation" Path
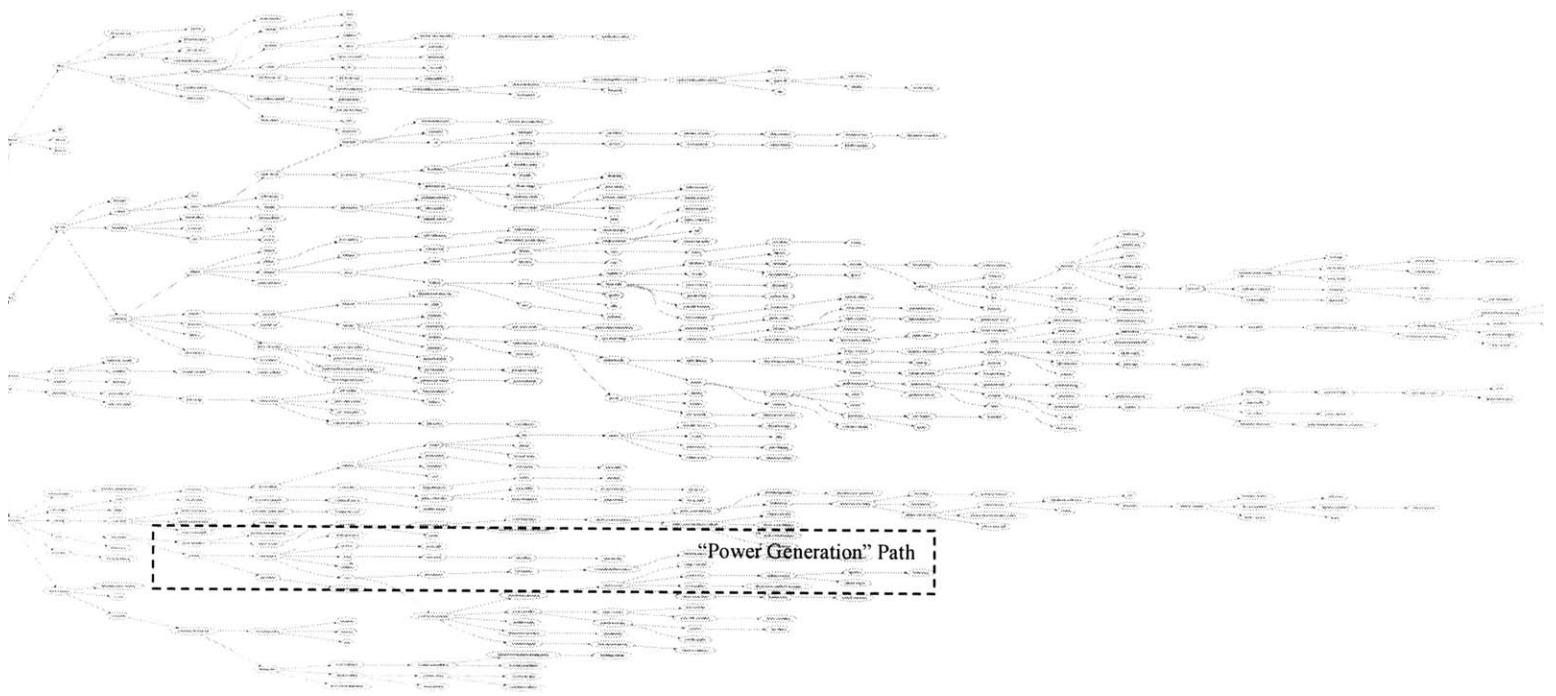
Figure 27: Visual Representation of DSC-Generated Taxonomy

CHAPTER 6: Conclusion

6.1 Recommendations

The previous chapter presented and analyzed the results generated by testing all the algorithm variants for taxonomy generation. In the end, we recommended the following two algorithm variants:

1. Heymann algorithm, cosine similarity, closeness centrality (H-CC)

2. DJP algorithm, symmetric NGD similarity, closeness centrality (D-SC)

6.2 Summary of Accomplishments

The overall goals of the project were outlined in Chapter 1 of this thesis. Here, the goals are recapped and a few quick notes are provided that outline our findings and how each goal was achieved.

1. To develop automated, publication database-independent methods for generating taxonomies.

The exact taxonomy generation process is introduced and discussed in detail in Chapter 3. We have developed software that first extracts terms from an online publication database such as Scopus or Engineering Village. The similarity between the terms chosen to be included in the taxonomy are quantified using either cosine, symmetric NGD, or asymmetric NGD similarity. After which the term similarities are stored in the form of a distance matrix. Finally, each taxonomy generation algorithm then uses this given distance matrix as a starting point to generate a taxonomy.

2. To compare several taxonomy generation algorithms and justify the usefulness of each.

The experiments conducted to test taxonomy generation algorithms are all introduced in Chapter 4 and the findings are presented in Chapter 5. Three sets of experiments were conducted. The first set tested for each taxonomy generation algorithm variant's consistency. The second set tested for each taxonomy generation algorithm variant's individual output's ability to conform to the distance matrix on which it was based. Finally, the third set created synthetic data sets of bibliometric information based on predetermined random taxonomies, attempting to simulate a typical set of documents we would expect to gather from a real online publication database, in order to test the ability of each taxonomy generation algorithm variant to reproduce underlying taxonomies.

3. To generate ways of visually representing the data in a manner that is easily understandable for viewers.

Chapter 3.6 describes the methods of visualizing the taxonomy. Specifically, the online ZGRViewer is presented that enables zooming and scrolling for efficient navigation within the taxonomy, allowing the user to focus on whatever parts of the taxonomy he wants to see.

4. To run a case study on "renewable energy.

All the results presented in Chapter 5 use the Scopus "renewable energy" data set, and taxonomies generated using the H-CC and D-SC taxonomy algorithm variants are presented in Chapter 5.5 using the most frequently occurring terms from the "renewable energy" data set.

6.3 Limitations and Suggestions for Further Research

The taxonomy generation algorithms developed and presented in this thesis have been rigorously tested, however there still remain two main areas where more research can be done. The reader is referred back to Figure 21 where the underlying model for taxonomy generation is presented. Specifically, the bottom row of the figure shows a clear path from the collection of bibliometric data from a publication database all the way to the generation of a taxonomy.

Firstly, experiments could be conducted using more publication databases. Currently, software has been developed for collecting information from Compendex, Inspec and Scopus, but more software can be developed for other publication databases in the future, and tests similar to those presented in Chapter 5 can be run.

Secondly, further investigation can be done concerning other similarity metrics used to generate distance matrices, as well as the advantages and disadvantages of each. The current similarity metrics used are cosine similarity, symmetric NGD similarity, and asymmetric NGD similarity.

There is a lot of ongoing work in the field of technology forecasting, and given the speed at which technology is progressing and the abundance of information present within media, lots of academic research is geared towards harnessing that information to aid technology forecasting. The work described in this thesis critically analyzes the process of generating visual representations of technology research landscapes through taxonomies. We believe that the work shown here is critical to technology forecasting as it helps researchers and decision-makers learn more about their field.

REFERENCES

[Blaschke 2002] Blaschke, C., Valencia, A. *Automatic Ontology Construction from the Literature*. Genome Informatics, Volume 13, 2002, pp. 201-213.

[Chuang et al. 2002] Chuang S., Chien L., *Towards Automatic Generation of Query Taxonomy: A Hierarchical Query Clustering Approach.* Academia Sinica, Taipei, 2002

[Cilibrasi & Vitanyi 2007] Cilibrasi R.L., Vitanyi P. M. B., *The google similarity distance.* IEEE Trans. Knowledge and Data Engineering Vol 19, Number 3, 2007, pp 370-383.

[Daim 2006] Daim, T.U., et al. *Forecasting emerging technologies: Use of bibliometrics and patent analysis.* Technological Forecasting and Social Change, Volume 73, Issue 8, 2006.

[Feldman 1998] Feldman, R., Dagan, I., Hirsh, H. *Mining Text Using Keyword Distributions.* Journal of Intelligent Information Systems, Volume 10, Issue 3, 1998.

[Firat et al. 2008] Firat, A., Woon W., Madnick S. *Technological Forecasting – A Review.* Composite Information Systems Laboratory (CISL), Massachusetts Institute of Technology, 2008.

[Glanzel 1996] Glanzel, W., *The need for standards in bibliometric research and technology.* Scientometrics, Akademiai Kiado, Volume 35, Number 2, 1996.

[Henschel at al. 2009] Henschel A., Woon W., Wachter, T., Madnick, S. *Comparison of Generality Based Algorithm Variants for Automatic Taxonomy Generation.* Composite Information Systems Laboratory (CISL), Massachusetts Institute of Technology, 2009.

[Heymann 2006] Heymann, P., Garcia-Molina, H., *Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems.* InfoLab Technical Report, Stanford University, 2006.

[Kostoff 2000] Kostoff, R., et al. *Fullerene Data Mining Using Bibliometrics and Database Tomography.* American Chemical Society, 2000.

[Kostoff 2001] Kostoff R., et al. *Citation Mining: Integrating Text Mining and Bibliometrics for Research User Profiling.* Journal of the American Society for Information Science and Technology, 2001.

[Krishnapuram 2003] Krishnapuram, R., Kimmamuru K., *Automatic Taxonomy Generation: Issues and Possiblities.* Lecture Notes in Computer Science, Springer, Berlin, 2003.

[Martino 1993] Martino, J. . *Technological Forecasting for Decision Making, 3rd ed.* Mc-Graw-Hill, New York, 1993.

[Martino 2003] Martino, J. *A Review of Selected Recent Advances In Technological Forecasting.* Technological Forecasting and Social Change. Vol 70, Number 8, October 2003, pp. 719-733.

[Narin 1996] Narin, F., Hamilton, K. *Bibliometric Performance Measures.* Scientometrics. Akademiai Kiado, Volume 36 Number 3, 1996.

[Okubo 1997] Okubo Y., *Bibliometric Indicators and Analysis of Research Systems: Methods and Examples.* OECD Science, Technology and Industry Working Papers, Number 1, 1997.

[Porter 1991] Porter, A., et al. *Forecasting and Management of Technology.* Wiley-Interscience, New York, 1991.

[Porter 2005] Porter, A., Cunningham S.. *Tech Mining.* Wiley-Interscience, New York, 2005.

[Porter 2007] Porter, A., *How "Tech Mining" Can Enhance R&D Management.* Research Technology and Management, Mar-Apr 2007.

[Sanchez 2004] Sanchez, D., Moreno, A., *Automatic Generation of Taxonomies from the WWW.* Practical Aspects of Knowledge Management, Volume 3336, 2004, pp 208-219.

[Schwarzkopf et al. 2007] Schwarzkopf, E., et al. *Mining the Structure of Tag Spaces for User Modeling.* Data Mining for User Modeling, International Conference on User Modeling, Greece, 2007

[Small 2006] Small, H., *Tracking and Predicting Growth Areas in Science.* Scientometrics, Akademiai Kiado, Hungary, 2006.

[van Raan 1996] van Raan, A.F.J., *Advanced Bibliometric Methods as Quantitative Core of Peer Review Based Evaluation and Foresight Exercises.* Scientometrics, Elsevier Science, Oxford, 1996.

[Verbeek 2002] Verbeek A., et al. *Measuring progress and evolution in science and technology – I: The multiple uses of bibliometric indicators.* International Journal of Management Reviews, Volume 4 Issue 2, 2002

[Vidican et al. 2009] Vidican G., Woon, W., Madnick, S. *Measuring Innovation Using Bibliometric Techniques: The Case of Solar Photovoltaic Industry.* Composite Information Systems Laboratory (CISL), Massachusetts Institute of Technology, 2009.

[Woon et al 2009(1)] Woon, W., Henschel, A., Madnick, S. *A Framework for Technology Forecasting and Visualization.* Composite Information Systems Laboratory (CISL), Massachusetts Institute of Technology, 2009

[Woon & Madnick 2008] Woon, W., Madnick. S. *Asymmetric Information Distances for Automated Taxonomy Creation.* Composite Information Systems Laboratory (CISL), Massachusetts Institute of Technology, 2008.

[Woon et al. 2009(2)] Woon W., Zeineldin, H., Madnick, S. *Bibliometric Analysis of Distributed Generations.* Composite Information Systems Laboratory (CISL), Massachusetts Institute of Technology, 2009.

[Ziegler 2009] Ziegler, B. *Methods for Bibliometric Analysis of Research: Renewable Energy Case Study.* Composite Information Systems Laboratory (CISL), Massachusetts Institute of Technology, 2009.

APPENDIX

Appendix A: Most Frequently Occurring Terms in Scopus "renewable energy" database

Terms with 500 or more occurrences in the database

solar-cell	23268 occurrences
wind powers	15776 occurrences
renewable energy resources	15175 occurrences
energy policies	14989 occurrences
fuel cells	13435 occurrences
photovoltaic cell	11865 occurrences
solar energy	8847 occurrences
biomass	8613 occurrences
mathematic models	7173 occurrences
computer simulator	6860 occurrences
electrical power generations	6216 occurrences
geothermal energy.	5958 occurrences
photovoltaic effect	5957 occurrences
wind turbines	5817 occurrences
energy efficiencies	5424 occurrences
electric generations	5397 occurrences
electrochemistry	5338 occurrences
thin film	5230 occurrences
electrical power system	4814 occurrences
hydrogen	4767 occurrences
energy resources	4755 occurrences
energy utilities	4549 occurrences
article	4403 occurrences
power generation	4339 occurrences
energy conservation	3988 occurrences
optimizing	3985 occurrences
biofuels	3925 occurrences
eurasia	3889 occurrences
electricity	3676 occurrences
carbon dioxide	3626 occurrences
energy conversion	3542 occurrences
environmental impacts	3500 occurrences
costs	3425 occurrences
silicon solar cells	3324 occurrences
electric utility	3147 occurrences

electrode	3115 occurrences
renewable resources	3078 occurrences
electrolyte	3040 occurrences
electrical potential	3040 occurrences
fossil fuels	3036 occurrences
sustainable development	2843 occurrences
solid-oxide fuel cell	2842 occurrences
oxidizers	2837 occurrences
electric batteries	2800 occurrences
energy management	2756 occurrences
methanol	2717 occurrences
economizers	2589 occurrences
catalysts	2587 occurrences
fuel	2584 occurrences
solar radiation	2533 occurrences
europe	2485 occurrences
carbon	2457 occurrences
environmental protection	2368 occurrences
marketing	2339 occurrences
biodiesel	2323 occurrences
ethanol	2267 occurrences
electric power distribution	2247 occurrences
catalysis	2213 occurrences
geothermal wells	2188 occurrences
oxygenates	2185 occurrences
priority journal	2171 occurrences
thermal effects	2142 occurrences
cost effectivity	2130 occurrences
electrical conductivities	2118 occurrences
silicon	2096 occurrences
nanostructured materials	2085 occurrences
polymers	2073 occurrences
semiconductive silicon	2062 occurrences
alternative energy	2060 occurrences
united states	2053 occurrences

solid-oxide fuel cell	2013 occurrences	algorithms	1499 occurrences
substrate	2010 occurrences	diesel fuel	1494 occurrences
heterojunctions	1997 occurrences	crystalline materials	1488 occurrences
current density	1997 occurrences	temperature	1459 occurrences
gas emission	1995 occurrences	titanium dioxide	1451 occurrences
climatic change	1989 occurrences	solar power generation	1447 occurrences
industrial economics	1914 occurrences	energy storage	1428 occurrences
technologies	1907 occurrences	vegetable oil	1422 occurrences
proton exchange membrane fuel cell	1894 occurrences	copper compounds	1422 occurrences
natural gas	1880 occurrences	wind-farm	1414 occurrences
turbining	1873 occurrences	developing country	1410 occurrences
investment	1873 occurrences	anodizing	1395 occurrences
renewable energy	1848 occurrences	solar equipment	1391 occurrences
solar cell array	1845 occurrences	north america	1387 occurrences
power plant	1838 occurrences	global warming	1378 occurrences
protonated	1834 occurrences	synthesis (chemical)	1376 occurrences
photovoltaic systems	1831 occurrences	computer networks	1363 occurrences
geothermal field	1829 occurrences	ester	1354 occurrences
diffusion	1826 occurrences	platinum	1340 occurrences
economic and social effects	1817 occurrences	deposition	1336 occurrences
amorphous silicon	1743 occurrences	performance	1335 occurrences
greenhouse effects	1726 occurrences	polyelectrolytes	1325 occurrences
laws and legislation	1718 occurrences	methanal	1313 occurrences
electric power plants	1677 occurrences	energy gaps	1311 occurrences
scanning electron microscopy	1675 occurrences	distributed generation	1302 occurrences
solar power plants	1617 occurrences	nonhuman	1290 occurrences
cathodes	1614 occurrences	ion exchange membranes	1288 occurrences
current-voltage characteristics	1593 occurrences	semiconductor materials	1283 occurrences
asia	1588 occurrences	water	1282 occurrences
air pollutants	1587 occurrences	strategic plan	1281 occurrences
geothermal systems	1566 occurrences	cell	1281 occurrences
direct energy conversion	1565 occurrences	light	1279 occurrences
biological materials	1561 occurrences	short circuit currents	1278 occurrences
project managers	1552 occurrences	annealing	1273 occurrences
combustion	1539 occurrences	reliable	1273 occurrences
electric inverters	1537 occurrences	crystals	1272 occurrences
heat	1536 occurrences	conversion efficiencies	1253 occurrences
electric currents	1534 occurrences	electric power transmission networks	1252 occurrences
energy consumption	1533 occurrences	asynchronous generators	1251 occurrences
powerful electronics	1523 occurrences	electricity industries	1251 occurrences
cell membranes	1521 occurrences	energy use	1245 occurrences
heat transfer	1506 occurrences	engine	1244 occurrences

dyeing	1218 occurrences
standards	1217 occurrences
models.	1216 occurrences
degradability	1213 occurrences
power transmission	1207 occurrences
fatty acid	1206 occurrences
semiconducting gallium arsenide	1202 occurrences
nuclear energy	1199 occurrences
photocurrents	1195 occurrences
power converters	1186 occurrences
interfaces (materials)	1179 occurrences
chemical vapor deposition	1177 occurrences
solar-powered	1175 occurrences
electrolysis	1174 occurrences
fermenter	1170 occurrences
concentration process	1165 occurrences
membranes	1158 occurrences
controller systems	1139 occurrences
international conferences	1137 occurrences
energy market	1137 occurrences
x-ray diffraction analysis	1137 occurrences
hydrocarbons	1132 occurrences
organic compounds	1131 occurrences
thermodynamics	1128 occurrences
diesel-engine	1124 occurrences
public policy	1122 occurrences
china	1122 occurrences
electrical load	1119 occurrences
wind-energy	1118 occurrences
doping (additives)	1106 occurrences
efficient	1100 occurrences
carbon monoxide	1099 occurrences
competition	1091 occurrences
sensors	1088 occurrences
cost-benefit analysis	1087 occurrences
voltage controllers	1085 occurrences
decision-making	1077 occurrences
coal	1067 occurrences
semiconducting films	1058 occurrences
optical properties	1046 occurrences
polycrystalline materials	1036 occurrences
esterification	1030 occurrences

light--absorption	1025 occurrences
forestry	1024 occurrences
x- ray diffractions	1018 occurrences
silicon wafers	1017 occurrences
electrochemical electrodes	1000 occurrences
thin film devices	998 occurrences
open circuit voltage	981 occurrences
pyrolysis	976 occurrences
reaction kinetics	961 occurrences
societies and institutions	955 occurrences
quantum efficiency	955 occurrences
biogas	951 occurrences
biotechnology	947 occurrences
hydroelectric power	946 occurrences
cellulose	943 occurrences
semiconducting cadmium compound	943 occurrences
electrons	939 occurrences
building	938 occurrences
emissions controls	937 occurrences
zinc oxide	932 occurrences
absorption	928 occurrences
parameters estimation	925 occurrences
semiconductor junctions	919 occurrences
synthetic fuels	919 occurrences
greenhouse gases	917 occurrences
alcoholate	917 occurrences
pump	914 occurrences
computer software	900 occurrences
metallation	900 occurrences
conservation	899 occurrences
adsorption	898 occurrences
electrical resistivity	892 occurrences
direct methanol fuel cells (dmfcs)	890 occurrences
photovoltaic	888 occurrences
silicas	887 occurrences
hydrolysis	882 occurrences
transmission electron microscopy	881 occurrences
heat pump systems	878 occurrences
geothermal power plants	873 occurrences
acidization	867 occurrences
plasma enhanced chemical	865 occurrences

vapor deposition	
fullerenes	862 occurrences
morphology	859 occurrences
geochemistry	858 occurrences
dye sensitive solar cell	854 occurrences
wind effects	854 occurrences
energy	853 occurrences
wind speed	850 occurrences
photoelectrochemical cells	844 occurrences
nonmetal	842 occurrences
gasoline	842 occurrences
glass	841 occurrences
heat exchange	839 occurrences
soil	835 occurrences
research and development management	833 occurrences
catalyst activations	831 occurrences
agriculture	826 occurrences
wood	826 occurrences
cytology	821 occurrences
economics--analysis	820 occurrences
far east	819 occurrences
photoluminescence	817 occurrences
glucose	817 occurrences
photonics	813 occurrences
charge transfer	812 occurrences
electric property	810 occurrences
hydrogen fueling	807 occurrences
mass transfer	806 occurrences
evaporation	806 occurrences
electric power utilization	803 occurrences
polymeric membranes	802 occurrences
passive	801 occurrences
bioenergy	797 occurrences
crude petroleum	796 occurrences
polarization	790 occurrences
desalination	786 occurrences
charge carriers	785 occurrences
photoconduction	781 occurrences
carbon emissions	780 occurrences
electric power transmission	776 occurrences
problem solving	775 occurrences
simulation results	774 occurrences

power control	772 occurrences
planning	769 occurrences
bioreactors	768 occurrences
solar concentration	768 occurrences
gas turbines	764 occurrences
wind-powered generators	760 occurrences
power system	759 occurrences
semiconductor doping	759 occurrences
semiconducting indium compounds	758 occurrences
gasification	757 occurrences
copper	753 occurrences
winds	750 occurrences
power qualities	749 occurrences
greenhouse gas	747 occurrences
nitrogenation	744 occurrences
forecasting.	741 occurrences
cooling	739 occurrences
reduction	736 occurrences
air pollution control	724 occurrences
nuclear power plants	723 occurrences
energy planning	723 occurrences
rural areas	721 occurrences
electric energy	720 occurrences
porosity	719 occurrences
ion-exchange	719 occurrences
financing	718 occurrences
chemical reaction	713 occurrences
organic polymers	711 occurrences
simulating	710 occurrences
electric network analysis	703 occurrences
renewable-energy sources	703 occurrences
leakage (fluid)	702 occurrences
designers	701 occurrences
life-cycle	699 occurrences
alternative fuels	694 occurrences
porous materials	689 occurrences
productivity	688 occurrences
data acquisition	687 occurrences
capacitors	686 occurrences
aerodynamics	685 occurrences
distributed power generations	684 occurrences
ecology	684 occurrences

composite	683 occurrences	comparative studies	613 occurrences
recycling	683 occurrences	satellites	612 occurrences
world	678 occurrences	geothermal heating	611 occurrences
particulate emissions	674 occurrences	aluminum	611 occurrences
biofuel production	674 occurrences	electrodeposit	610 occurrences
controlled study	672 occurrences	glycerol	610 occurrences
eirev	666 occurrences	electric power supplies to apparatus	608 occurrences
electric converters	665 occurrences	hydrogen productivity	607 occurrences
reviews	662 occurrences	semiconductor devices	606 occurrences
enzymes	661 occurrences	systems analysis	604 occurrences
western europe	659 occurrences	installation	604 occurrences
regulatory compliance	657 occurrences	data reduction	603 occurrences
speed	657 occurrences	reactive power	602 occurrences
india	657 occurrences	spacecraft	602 occurrences
sensitivity analysis	656 occurrences	hydraulic motor	602 occurrences
microstructure	654 occurrences	geothermal reservoir	600 occurrences
taxation	652 occurrences	electric powers	598 occurrences
geothermal prospect	652 occurrences	sulfur	598 occurrences
gases	651 occurrences	statistical methods	598 occurrences
geological	651 occurrences	sensor networks	597 occurrences
irradiance	650 occurrences	chemistry	594 occurrences
united kingdom	645 occurrences	crops	594 occurrences
single crystals	645 occurrences	water supplies	594 occurrences
geophysical	643 occurrences	fabric	590 occurrences
humans	641 occurrences	electric energy storage	590 occurrences
grain boundaries	638 occurrences	cadmium compounds	590 occurrences
x ray photoelectron spectroscopy	636 occurrences	gas industry	587 occurrences
film growth	636 occurrences	petroleum industries	584 occurrences
pem-fuel cell	636 occurrences	bioelectric energy sources	582 occurrences
fuel consumption	635 occurrences	groundwater	582 occurrences
nanoparticle	635 occurrences	spectrum analysis	579 occurrences
transesterification	633 occurrences	geophysics	geothermal
technology transfer	627 occurrences	deregulation	578 occurrences
methodology	627 occurrences	geothermal gradient	578 occurrences
crystals--structure	621 occurrences	wastewater - treatment	578 occurrences
risk assessment	619 occurrences	electric load forecasting	577 occurrences
hydraulic machinery	618 occurrences	neural networks	577 occurrences
flow of fluids	617 occurrences	atmospheric humidity	575 occurrences
methanol fuel	617 occurrences	electric supplies	574 occurrences
solar system	615 occurrences	international trade	572 occurrences
surface properties	615 occurrences	energy sourcing	571 occurrences
volcano	614 occurrences	wind generator	570 occurrences

synchronous generators	570 occurrences
biodegradable	570 occurrences
energy supply	569 occurrences
solar cell silicon	568 occurrences
wind velocities	567 occurrences
nanotechnologies	566 occurrences
cogeneration plants	566 occurrences
light-emitting diodes	564 occurrences
lignin	563 occurrences
ecosystems	563 occurrences
resource allocation	562 occurrences
carbon nanotube	561 occurrences
electric power factor	561 occurrences
probable	555 occurrences
technical presentations	554 occurrences
spectroscopic analysis	553 occurrences
germany	551 occurrences
electrical vehicles	550 occurrences
etch	550 occurrences
sintering	550 occurrences
wireless telecommunication systems	549 occurrences
capacitance	547 occurrences
atomic-force microscopies	546 occurrences
rotors	546 occurrences
southern europe	545 occurrences
numerical methods	545 occurrences
zea mays	544 occurrences
fuel economy	543 occurrences
composite materials	542 occurrences
system stabilization	541 occurrences
hydroelectric power plants	540 occurrences
semiconducting silicon compounds	539 occurrences
solute	539 occurrences
sustainable	535 occurrences

technology forecasts	534 occurrences
energy productivity	533 occurrences
nickelates	532 occurrences
potential energy	531 occurrences
wireless sensor network	530 occurrences
fourier-transform infrared spectroscopies	530 occurrences
computation methods	527 occurrences
wastewater	526 occurrences
heating treatments	524 occurrences
carriers concentration	524 occurrences
european unions	523 occurrences
waste management	523 occurrences
grid-connected	522 occurrences
energy demand	521 occurrences
thin film solar cell	521 occurrences
numeration model	521 occurrences
computational fluid dynamics	520 occurrences
stoichiometry	520 occurrences
environmental engineering	519 occurrences
activation energies	515 occurrences
zirconia	515 occurrences
power density	515 occurrences
ph	514 occurrences
pressure effects	513 occurrences
climatology	513 occurrences
heat-flow	512 occurrences
air quality	511 occurrences
aquifer	509 occurrences
optoelectronic device	507 occurrences
metabolism	506 occurrences
solar collectors	502 occurrences
electric power systems-- interconnection	500 occurrences

Appendix B: Masdar Initiative

This project described here is funded by the Masdar Initiative, founded in April 2006, which is a multi-faceted framework dedicated to advancing the development, commercialization and deployment of renewable energy solutions and clean technologies. The goal of Masdar is to integrate the full renewable and clean technology lifecycle - from research to commercial deployment – with the aim of creating scalable clean energy solutions. Masdar works with global partners and institutions to integrate new research with proven technologies to produce efficient systems and processes that can be replicated globally. [taken from www.masdar.ae] One goal is the construction of Masdar City, a zero carbon, zero waste city being built just outside Abu Dhabi, UAE. This site is the location of the Masdar Institute of Science and Technology. Masdar has a very clear interest in discovering more about the "renewable energy" field. The goal of our group's research is to provide our colleagues in Masdar with an automated way of discovering more about "renewable energy". All the case studies and results generated within this thesis all focus on "renewable energy" related technologies.

Appendix C: Description of Code

This section of the Appendix explains the different classes of python code used for this project. Classes arranged alphabetically.

ANALYSIS.py
****

This is the file that puts all the other files together. It enables the creation of a local data set and distance_matrices, and the running of algorithms to produce a taxonomy.
No methods are included in this file.

compare.py
****

This file compares two taxonomies. Taxonomies are always saved in the form of a *Taxonomy* object as defined in HEYMANN.py
This file has a dependency on HEYMANN.py
Methods in this file include:
```
compareTaxonomySameTermlist(taxonomy_file1, taxonomy_file2, scoringScheme, emphasis)
compareTaxonomyDiffTermlist(taxonomy_file1, taxonomy_file2,scoringScheme, emphasis)
analyzeTaxonomy(taxonomy_file)
```


database.py
****

This file creates an SQLite3 database of bibliometric information from Compendex / Inspec, splits an SQLite3 database from Compendex / Inspec (ev) and Scopus.
Methods in this file include:
```
create_ev_db(seed_terms, db_name, num_abstracts, db_number, next_url)
createSqlDb(db_name)
split_database(orig_dbfilename, num_docs, new_dbfilename, source)
```


figure.py
****

This file generates figure given a connection matrix. It uses the pydot python package.
Methods in this file include:
```
generate_figure(connections, keywords, filename, distmat)
```


keywords.py
****

This file creates keyword/term lists and scans them.
Methods in this file include:
```
stemphrase(phrase, lowercase)
get_keywords(num_keywords, dbfilename, source, keywords_file, keywordsFromScratchBoolean)
print_keywords(keyword_file, num_keywords, limit)
compare_keywords(keyword_file1, keyword_file2)
```

distance_matrix.py
****

This file generates all the tags in the database for a set of terms in the term list. It then uses the tags to generate the cosine, symmetric and asymmetric NGD distance matrices.
Methods in this file include:

```
generate_tags(termlist, dbfilename, source)
generateDistanceMatrix(termlist, dbfilename, source, distanceMatrix_file)
getDistanceMatrix(distanceMatrix_file, distanceMatrix_type)
```

score.py
****

This file scores taxonomies that are in the form of connection matrices
Methods in this file include:

```
score_taxonomy_average(connmat, dirdistmat, roots=[])
score_taxonomy_momentum(connmat, dirdistmat, roots=[])
score_taxonomy_weighted(connmat, dirdistmat, roots=[], weighting=0)
```

toy_data.py
****

This file generates a random taxonomy and a set of documents that attempt to 'realistically' reflect the taxonomy. The documents are saved in an SQLite3 database. The idea here is to try to reconstruct the taxonomy by feeding the `toy_data.py`'s SQLite3 database as input and running one of the taxonomy generation algorithms.
This file has a dependency on `HEYMANN.py`
Methods included in this file are:

```
generateDatabase(num_terms, dbfilename, outputFilename, kwPerDoc, docsPerTerm,
mainTermWeight = 0.4, num_suppTerms = 5, suppTermWeight = 0.1)
```

KRUSKALS.py
****

This file uses Kruskal's algorithm to generate taxonomy
Methods included in this file are:

```
create_taxonomy_kruskals(distance_matrix, root=0)
```

DJP.py
******

This file uses Dijsktra-Jarnik-Prim's (DJP) algorithm to generate taxonomy.
Methods included in this file are:

```
create_taxonomy_djp(distance_matrix, root=0)
```

EDMONDS.py
****

This file uses Edmond's algorithm to generate taxonomy.
Methods included in this file are:

```
create_taxonomy_edmonds(directional_distmat, root=0)
```

HEYMANN.py
****

This file generates taxonomy using Heymann algorithm. It also constructs *Taxonomy* objects, which are made up of *Tag* objects where each *Tag* represents a term in the taxonomy.
Classes included in this file are:

Tag
- Attributes: name, dmIndex, similarityList, parent, centrality, distribution
- Methods within this class:

```
similarity(tag)
```

Taxonomy
- Attributes: root, tree, vertices, vertexDict
- Methods:

```
children(tagname)
ancestors(tag)
addVertex(parent, vertex)
_addVertex(vertex)
addEdge(parent, vertex)
findMostSimilar(tag, goal="min", nonEligibleTags=set())
hasEdge(parent, child)
hasGrandchild(parent, child, maxlevel=2, level=0)
simpleDraw(root=[], indent)
draw(filename)
treeToMatrix(termlist)
compare(taxonomy2, scoringScheme, emphasis)
```

HeymannAlgorithm
- Attribute: tags
- Methods:

```
centralityOrder(centrality="closeness", kind="directed"),
run(taxThreshold, centrality, kind, goal, rerunOnRemaining)
```

Other methods included in this file are:

```
addWeight(val, level, depth, scoringScheme, emphasis)
matrixToTree(connmat, termlist)
```

GENETIC.py
****

This file generates taxonomy using Genetic algorithm.
Methods included in this file are:

```
find_descendants(connmat, node)
random_taxonomy(num_nodes, root[])
crossover_taxonomy(connmat1, connmat2, root=[])
mutate_taxonomy(connmat)
create_taxonomy_genetic(directional_distmat, dummy_distmat=[], root=[], num_iterations,
num_chromosomes, num_fittest, num_mutations, num_crossovers, cost_function,
initial_chromosomes)
```

GUI.py
****

This file creates the graphic user interface for taxonomy generation.

Appendix D: The Graphic User Interface

A simple graphic user interface was developed using python's Tkinter package.

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit and a de-facto standard GUI in general, and is included with the standard Windows install of Python. Tk is an open source, cross-platform widget toolkit – that is, a library of basic elements for building a graphical user interface (GUI). Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.[11]

This section describes the graphic user interface (GUI) developed for taxonomy generation. A screenshot of the interface is shown below and can be accessed by opening the GUI.py file in the source code.
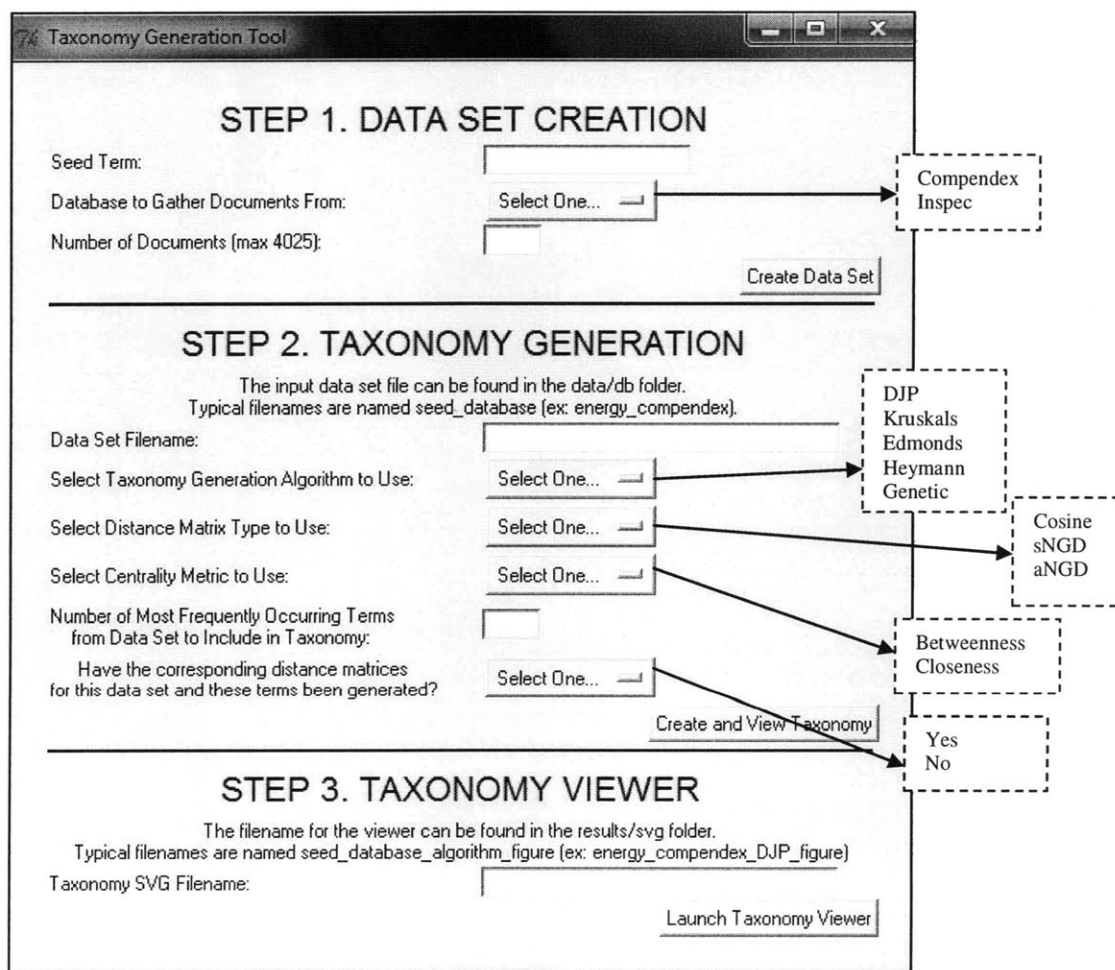
Graphic User Interface for Taxonomy Generation Screenshot
The Choices for each dropdown box are indicated in the figure

As can be seen, the interface is divided into three distinct subsections.

---

[11] Taken from http://en.wikipedia.org/wiki/Tkinter

The top section labeled "Step 1: Data Set Creation" generates the data set containing relevant bibliometric information from an online publication database. It has text fields for entering the query term (*seed term*) to be entered into the online publication database's search box, and the number of documents to be collected from the database. It also has a dropdown box indicating which online publication database to collect information from. Currently, it is only configured to work for Compendex and Inspec, both databases under the Engineering Village umbrella, hence there is a limitation that only up to the first 4,025 relevant documents can be gathered. Once the "Create Data Set" button is clicked, popup alert boxes alert the user of the data collection process' progress. If the user fails to complete all the fields / parameters necessary, a popup alert box will appear alerting the user of this failure as well.

Before gathering any bibliometric information using this GUI, it is recommended that the user visit http://www.engineeringvillage.com first to ensure that there are a sufficient number of documents that can be collected from Compendex / Inspec using the inputted seed term. Sometimes, inputting a certain seed term will produce fewer than 4,025 resulting documents, in which case the user cannot collect the bibliometric information for that many documents.

The middle section labeled "Step 2: Taxonomy Generation" generates the actual visual representation of the taxonomy. It contains fields for entering the name of the data set file and number of terms to be included in the taxonomy, as well as several dropdown boxes to choose which exact taxonomy generation algorithm to use. Clicking on the "Create and View Taxonomy" button generates the taxonomy based on the given parameters and shows the taxonomy viewer in the taxonomy viewer. If there are any errors in the way the fields in this subsection were completed, popup alert boxes will appear, notifying the user.

Finally, the lowest section labeled "Step 3: Taxonomy Viewer" loads a saved taxonomy into a taxonomy viewer, which opens in a web browser window.

Appendix E: Tests for Engineering Village

As a further test, a backend data set composed of Engineering Village-collected documents was used and the same tests as discussed in Chapter 5 (Results) were run to further solidify the claims stated in Chapter 5 and 6. Note however, that the collected data set from Engineering Village was small, containing only 23,048 documents gathered from Compendex, which is a whole order of magnitude less than the Scopus data set of 153,537 documents. As such, the results are not as accurate as the results generated and shown in Chapter 5, but are nevertheless presented here anyway.

Evaluating the Consistency of Taxonomy Generation Algorithms

The first set of tests were aimed at evaluating the consistency of the taxonomy generation algorithms. To do this, two sets of experiments were conducted to gauge the consistency, or robustness against noise, of the different taxonomy generation algorithms. The first set of experiments measured the consistency of the algorithms with slight perturbations in the backend, while in the second set of experiments, the backend database was fixed and perturbations were introduced to the collection of terms used to form the taxonomy.

Backend Data Set Consistency

For this test, the 23,048-entry bibliometric data set is randomly divided into five separate 15,000-entry subsets. The most popular terms from the entire Compendex "renewable energy" bibliometric data set were then taken and each of the taxonomy generation algorithms were run, keeping constant the term list and varying the backend data set between the five 15,000-entry sets. The percentage similarity of direct links between each of the taxonomies generated was then calculated between each of the 15,000-entry-backend data set taxonomies and the entire 23,048-entry-backend data set taxonomy. The table below summarizes the mean of the percentage similarities for each algorithm variant.

Backend Data Set Consistency Test Results

Algorithm Variant Acronym	25 most frequently occurring terms used as term list	50 most frequently occurring terms used as term list	100 most frequently occurring terms used as term list	Mean of Percentage Similarities
D-CB	92.00%	94.40%	93.40%	93.27%
D-CC	100.00%	94.40%	93.40%	95.93%
D-SB	90.40%	81.20%	86.20%	85.93%
D-SC	90.40%	86.80%	89.40%	88.87%
K-CB	92.00%	94.40%	93.40%	93.27%
K-CC	100.00%	94.40%	93.40%	95.93%
K-SB	4.00%	1.20%	0.40%	1.87%

K-SC	4.00%	1.60%	1.00%	2.20%
E-AB	95.20%	93.20%	94.80%	94.40%
E-AC	95.20%	93.20%	94.80%	94.40%
H-AB	97.60%	98.40%	97.40%	97.80%
H-AC	97.60%	**98.80%**	**99.20%**	**98.53%**
H-CB	32.80%	58.80%	53.40%	48.33%
H-CC	98.40%	93.60%	94.20%	95.40%
H-SB	67.20%	53.20%	74.80%	65.07%
H-SC	91.20%	87.20%	89.00%	89.13%

Highlighted in the table above are the top performers for each test run. Based on these results, the best performing algorithm variants (over 95% similarity on average) are:

DJP algorithm, cosine similarity, closeness centrality for root selection (D-CC)

Kruskals algorithm, cosine similarity, closeness centrality for root selection (K-CC)

Edmonds algorithm, asymmetric NGD similarity, betweenness centrality (E-AB)

Edmonds algorithm, asymmetric NGD similarity, closeness centrality (E-AC)

Heymann algorithm, asymmetric NGD similarity, betweenness centrality (H-AB)

Heymann algorithm, asymmetric NGD similarity, closeness centrality (H-AC) – **top performer!**

Heymann algorithm, cosine similarity, closeness centrality (H-CC)

Term Consistency

For this test, the backend was kept constant, and consisted of the entire 23,048-entry Compendex "renewable energy" bibliometric data set. However, the term lists were varied by taking the most popular terms in the data set and inserting "noise" terms, which are terms selected randomly from the rest of the terms in the data set. We chose to insert an equal number of noise terms to the terms already in the taxonomy. For instance, if a taxonomy was created using the 25 most frequently occurring terms, 25 noise terms were inserted into the taxonomy, then each taxonomy generation algorithm was run using those 50 total terms, and percentage of the number of links consistent in the 25-term noise-free and 50-term noisy taxonomies outputted by each taxonomy generation algorithm was calculated. This test was repeated three times and the mean of the three percentage link similarities for each algorithm was taken. The results are summarized in the table on the next page.

Term Consistency Test Results

Algorithm Variant Acronym	25 most frequently occurring terms, with 25 more noise terms	50 most frequently occurring terms, with 50 more noise terms	100 most frequently occurring terms, with 100 more noise terms	Mean of Percentage Similarities
D-CB	91.03%	94.77%	91.09%	92.30%
D-CC	**96.15%**	94.77%	91.09%	94.00%
D-SB	60.26%	78.43%	88.12%	75.60%
D-SC	80.77%	82.35%	88.45%	83.86%
K-CB	91.03%	94.77%	91.09%	92.30%
K-CC	**96.15%**	94.77%	91.09%	94.00%
K-SB	0.00%	0.65%	0.00%	0.22%
K-SC	2.56%	0.00%	0.99%	1.18%
E-AB	**96.15%**	94.12%	94.72%	95.00%
E-AC	**96.15%**	94.12%	94.72%	95.00%
H-AB	94.87%	**96.73%**	96.37%	95.99%
H-AC	**96.15%**	96.08%	**97.69%**	**96.64%**
H-CB	43.59%	59.48%	43.89%	48.99%
H-CC	93.59%	88.89%	89.44%	90.64%
H-SB	29.49%	41.83%	57.43%	42.91%
H-SC	71.79%	67.32%	68.32%	69.14%

Highlighted in the table above are the top performers for each test run. The best performing algorithm variants (over 90% similarity) based on our tests are:

DJP algorithm, cosine similarity, betweenness centrality for root selection (D-CB)

DJP algorithm, cosine similarity, closeness centrality for root selection (D-CC)

Kruskals algorithm, cosine similarity, betweenness centrality for root selection (K-CB)

Kruskals algorithm, cosine similarity, closeness centrality for root selection (K-CC)

Edmonds algorithm, asymmetric NGD similarity, betweenness centrality (E-AB)

Edmonds algorithm, asymmetric NGD similarity, closeness centrality (E-AC)

Heymann algorithm, asymmetric NGD similarity, betweenness centrality (H-AB)

Heymann algorithm, asymmetric NGD similarity, closeness centrality (H-AC) – **top performer!**

Heymann algorithm, cosine similarity, closeness centrality (H-CC)

Consistency Test Summary

The consistency tests were run both by varying the backend data set and term lists to test for taxonomy generation algorithm robustness versus noise. The table below repackages the information from the previous two tables for easier viewing.

Consistency Test Summary

Algorithm Variant Acronym	Mean of Percentage Similarities for Backend Data Set Consistency Test	Mean of Percentage Similarities for Term List Consistency Test
D-CB	93.27%	92.30%
D-CC	95.93%	94.00%
D-SB	85.93%	75.60%
D-SC	88.87%	83.86%
K-CB	93.27%	92.30%
K-CC	95.93%	94.00%
K-SB	1.87%	0.22%
K-SC	2.20%	1.18%
E-AB	94.40%	95.00%
E-AC	94.40%	95.00%
H-AB	97.80%	95.99%
H-AC	**98.53%**	**96.64%**
H-CB	48.33%	48.99%
H-CC	95.40%	90.64%
H-SB	65.07%	42.91%
H-SC	89.13%	69.14%

The most consistent algorithms variants are D-CC, K-CC, E-AB, E-AC, H-AB, H-AC and H-CC, and the top performer was H-AC.

Evaluating Individual Taxonomies

Several tests were run which tested each of the taxonomy generation algorithms' outputs individually by taking their outputs and scoring them using the different scoring metrics described in the previous chapter. To recap, the scoring metrics used were (for more information about each of the metrics mentioned above, see the Chapter 4 of this thesis):

1. Average
2. Momentum
3. Mean to Root
4. Mean to Grandparent
5. Linear
6. Exponential (0.5)
7. Exponential (0.75)

Note that the scoring algorithms measure each taxonomy's conformity to its distance matrix and as such are only useful when comparing taxonomies generated using the same similarity metric since only one similarity metric characterizes a distance matrix. This means that using a given scoring metric, it is impossible to compare all the taxonomy generation algorithms to each other, however it is possible to compare all the taxonomy generation algorithms that used the cosine similarity metric, symmetric NGD similarity metric, or asymmetric NGD similarity metric to each other.

The top 100, 250 and 500 frequently occurring terms in the Compendex "renewable energy" data set were used in conjunction with the entire bibliometric data set . The results are presented in the following subsections.

Using the top 100 terms

The results summarized in the table below are from tests run using the cosine similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Different Scoring Metrics used on Cosine Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-CB	0.336	28.877	26.848	28.877	28.759	29.492	27.994
D-CC	0.336	28.877	26.848	28.877	28.759	29.492	27.994
K-CB	0.336	28.877	26.848	28.877	28.759	29.492	27.994
K-CC	0.336	28.877	26.848	28.877	28.759	29.492	27.994
H-CB	0.301	26.116	24.958	26.116	26.357	26.874	25.767
H-CC	0.330	28.800	26.733	28.800	28.608	29.275	27.838

The results summarized in the table below are from tests run using the symmetric NGD similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Different Scoring Metrics used on Symmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-SB	0.097	11.790	16.611	11.790	15.017	12.371	14.381
D-SC	0.097	11.744	15.758	11.744	14.330	12.197	13.916
K-SB	0.185	19.153	19.153	19.153	18.938	18.938	19.061

K-SC	0.185	18.461	18.461	18.461	18.477	18.477	18.468
H-SB	0.104	12.965	15.722	12.965	14.279	13.201	14.580
H-SC	0.100	11.994	**14.827**	11.994	**13.499**	12.331	**13.642**

Finally, the results summarized in the table below are from tests run using the asymmetric similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Different Scoring Metrics used on Asymmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
E-AB	**0.019**	**2.614**	**2.617**	**2.614**	**2.372**	**2.371**	**2.511**
E-AC	**0.019**	**2.614**	**2.617**	**2.614**	**2.372**	**2.371**	**2.511**
H-AB	**0.019**	2.626	2.631	2.626	2.386	2.381	2.524
H-AC	**0.019**	2.619	2.628	2.619	2.384	2.378	2.521

Using the top 250 terms

The results summarized in the table below are from tests run using the cosine similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Different Scoring Metrics used on Cosine Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-CB	**0.339**	**69.153**	54.866	**69.153**	61.843	**68.556**	60.878
D-CC	**0.339**	**69.153**	54.866	**69.153**	61.843	**68.556**	60.878
K-CB	**0.339**	**69.153**	54.866	**69.153**	61.843	**68.556**	60.878
K-CC	**0.339**	**69.153**	54.866	**69.153**	61.843	**68.556**	60.878
H-CB	0.299	59.077	51.515	59.077	57.388	60.722	55.417
H-CC	0.332	67.705	**55.657**	67.705	**62.819**	67.629	**60.922**

The results summarized in the table below are from tests run using the symmetric NGD similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Different Scoring Metrics used on Symmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-SB	0.103	31.854	68.249	31.854	56.021	35.180	44.958
D-SC	0.103	31.854	52.146	31.854	48.131	35.183	43.486
K-SB	0.234	73.524	73.524	73.524	68.502	68.502	71.372
K-SC	0.234	57.826	57.826	57.826	58.037	58.037	57.916
H-SB	0.111	35.591	54.145	35.591	45.898	38.444	46.156
H-SC	0.108	33.361	50.911	33.361	45.807	36.094	43.526

Finally, the results summarized in the table below are from tests run using the asymmetric NGD similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Different Scoring Metrics used on Asymmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
E-AB	0.029	10.136	11.021	10.136	9.647	9.529	10.363
E-AC	0.029	10.136	11.021	10.136	9.647	9.529	10.363
H-AB	0.029	10.190	11.051	10.190	9.681	9.566	10.397
H-AC	0.029	10.174	11.037	10.174	9.663	9.545	10.380

Using the top 500 terms

The results summarized in the table below are from tests run using the cosine similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Different Scoring Metrics used on Cosine Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-CB	0.351	139.653	89.393	139.653	105.257	135.146	111.742

D-CC	**0.351**	**139.653**	89.393	**139.653**	105.257	135.146	111.742
K-CB	**0.351**	**139.653**	89.397	**139.653**	105.262	**135.147**	111.745
K-CC	**0.351**	**139.653**	89.397	**139.653**	105.262	**135.147**	111.745
H-CB	0.304	118.458	94.774	118.458	109.421	119.117	105.292
H-CC	0.335	135.360	**102.326**	135.360	**119.557**	132.933	**116.100**

The results summarized in the table below are from tests run using the symmetric NGD similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Different Scoring Metrics used on Symmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
D-SB	**0.102**	64.531	174.648	64.531	145.793	74.442	103.011
D-SC	**0.102**	**64.013**	121.616	**64.013**	109.554	**73.119**	95.333
K-SB	0.259	154.430	154.430	154.430	146.130	146.130	150.873
K-SC	0.259	134.702	134.702	134.702	132.978	132.978	133.963
H-SB	0.110	73.751	124.443	73.751	102.357	82.680	103.551
H-SC	0.106	67.310	**104.232**	67.310	**92.437**	73.680	**89.190**

Finally, the results summarized in the table below are from tests run using the asymmetric NGD similarity metric to generate the distance matrix. Highlighted are the best taxonomy generation algorithms for each scoring metric.

Different Scoring Metrics used on Asymmetric NGD Similarity based Taxonomy Generation Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grandparent	Linear	Exponential (0.5)	Exponential (0.75)
E-AB	**0.026**	19.913	22.429	19.913	19.059	18.690	20.778
E-AC	**0.026**	19.913	22.429	19.913	19.059	18.690	20.778
H-AB	**0.026**	**19.825**	**22.374**	**19.825**	19.018	**18.633**	**20.716**
H-AC	**0.026**	19.879	22.376	19.879	**19.016**	18.636	20.719

Evaluating Individual Taxonomies Analysis

The consistently top-scoring algorithms among the 100, 250 and 500 term list tests are summarized in the table below. The shaded cells represent the consistently top-scoring algorithm variants for each of the scoring metrics.

Consistently Top Scoring Algorithm Variants

Algorithm Variant Acronym	Average	Momentum	Mean To Root	Mean To Grand-parent	Linear	Exponential (0.5)	Exponential (0.75)
D-CB	▓	▓					
D-CC	▓	▓					
D-SB	▓						
D-SC	▓	▓		▓			
K-CB	▓					▓	
K-CC	▓					▓	
K-SB							
K-SC							
E-AB	▓						
E-AC	▓	▓					
H-AB	▓						
H-AC	▓						
H-CB							
H-CC							
H-SB							
H-SC			▓		▓		

Based on the data in the table above, the algorithm variant that performed the best is:

DJP algorithm, symmetric NGD similarity, cosine centrality for root selection (D-SC)

The top performing algorithm variants in the consistency tests and the individual taxonomy tests using the Engineering Village backend data set are consistent with the results presented in Chapter 5 using the Scopus backend data set. The top performing algorithm variants using the Scopus backend also performed well using the Engineering Village backend.