A REALM-BASED QUESTION ANSWERING SYSTEM USING PROBABILISTIC MODELING

By

CLINT PAZHAYIDAM GEORGE

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2010

To my parents, George and Gracy Pazhayidam

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. Joseph N. Wilson, for all of the guidance and encouragement he provided me throughout my masters course work and research. I also thank my thesis committee members, Dr. Paul Gader, Dr. Sanjay Ranka, for all their help and suggestions.

In addition, I am grateful to my colleagues Joir-dan Gumbs, Guillermo Cordero, Benjamin Landers, Patrick Meyer, Christopher Shields, and Terence Tai for their assistance in building this system. I am particularly thankful to Peter J. Dobbins and Christan Grant for their insightful comments and ideas.

I thank my parents George and Gracy Pazhayidam and my family, Christa, Renjith, and Chris for their everlasting love, care, and encouragement, which motivated me throughout my studies. Finally, I offer my regards to all of my friends who helped me on the completion of the thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

A REALM-BASED QUESTION ANSWERING SYSTEM USING PROBABILISTIC MODELING

By

Clint Pazhayidam George

December 2010

Chair: Joseph N. Wilson
Major: Computer Engineering

Conventional search engines perform key-word based searches of web documents. Since the wealth of information from the web is huge and often represented in an unstructured format, the availability of automated search engines that can solve user queries has become essential. This thesis discusses categorizing and representing the information contained within web documents in a more structured and machine readable form, an ontology, and building a realm-based question answering (QA) system that uses deep web sources by exploiting information from the web along with sample query answering strategies provided by users.

The first part of the thesis deals with probabilistic modeling of the web documents using a Bayesian framework, Latent Dirichlet Allocation (LDA), and the corresponding document classification framework built on top of this. Subsequently, LDA is employed as a dimensionality reduction tool for the corpus vocabulary and documents. This thesis also describes the inference of document-topic-proportions for newly encountered documents without retraining the previously learned LDA model. The second part of the thesis describes a realm based QA system using an ontology that is built from the web. Finally, the techniques to represent user-queries that are natural language text in a semi-structured format (SSQ) and to rank the similar queries using a novel ontology based similarity quasi-metric, class divergence, is explained.

CHAPTER 1
INTRODUCTION

Conventional search engines are mainly focused on key-word based search in the web. So if we use them to answer a question, they usually give us relevant page links based on the key words in the query. To reach a web page providing a relevant answer to the query, we may have to follow several links or pages. Since the web is a huge database of information, which is often stored in an uncategorized and unstructured format, an automated system that can solve natural language questions has become essential. Current question answering (QA) systems attempt to deal with documents varying from small local collections (*closed domain*) to the World Wide Web (*open domain*)[16, 22]. This thesis discusses Morpheus [9], a QA system that is open domain and focused on a realm-based QA strategy.

Morpheus provides web pages offering a relevant answer to a user query by re-visiting relevance-ranked prior search pathways. Morpheus has two types of users: one is a *path finder* who searches for the answers in the web and records the necessary information to revisit the search pathways to the answer. The other type is a *path follower*, who enters a query to the system and gets an answer. Like many QA systems, Morpheus parses natural language queries and stores them in a semi-structured format, a semi-structured query or SSQ, that contains query terms, assigned term classes, and information about prior search paths. The term classes are drawn from an ontology and they are used to rank the similarity of prior search queries to a newly encountered query. This thesis focuses on modeling and categorizing web documents in order to help automate the ontology building process from the web. This thesis also describes ranking of the user queries in the SSQ format based on a quasi-metric, class divergence, defined on the ontology heterarchy.

Morpheus's QA follows a hierarchical approach, i.e, it categorizes user queries into a realm and solves the QA problem under that sub-domain or realm. A realm of information in the web is represented by an ontology that contains categories or classes and their

relationships. Morpheus's NLP engine tags the query terms with these classes and clusters queries based on the term classes' cumulative divergence calculated from the from the ontology. In this way, we can rank the similar solved queries stored in the database and reuse them for future searches.

Categorizing and extracting information from text is the main step in an ontology learning from web documents. This thesis describes a well known topic modeling method, Latent Dirichlet Allocation [6], and its applications in document modeling, dimensionality reduction, classification, and clustering.

Thesis Organization

The contribution of this thesis is a principled development of a probabilistic framework for information extraction and document classification to help the ontology building process, and a method for ranking prior search queries based on a hierarchy based similarity measure[9]. This thesis is organized as follows:

- Chapter 2 discusses the background for this research such as QA systems in general and the Morpheus system in particular. It also addresses the necessity of an ontology in the Morpheus system and gives an introduction to conventional ontology generating systems. In the end, this chapter shows different document modeling strategies and the motivation behind pursuing LDA for web document modeling in this thesis.

- Chapter 3 starts by describing the graphical model of LDA. Then, it explains feature extraction steps and topic extraction from web documents using LDA. It also discusses LDA based dimensionality reduction techniques, document classification, and the inference of document topic mixtures without retraining the learned LDA model.

- Chapter 4 presents the problem of ranking the prior search queries stored in the SSQ format, based on a novel class divergence measure that is built on the ontological heterarchy.

- Chapter 5 summarizes the idea of this thesis and discusses the directions of future work.

CHAPTER 2
BACKGROUND

The goal of this chapter is to give an overview of question answering (QA) systems in general and the Morpheus system in particular. It also looks into the need for an ontology and corpora in Morpheus system and their specifications. Subsequently, it describes some of the existing ontologies and ontology building systems such as DBpedia, Word Net, and YAGO and limitations of their ontologies and methodologies in the scope of Morpheus. This chapter concludes by discussing applicability of a well-known topic modeling technique Latent Dirichlet Allocation in web page categorization and how it aids the ontology and associated corpora building process.

2.1 Question Answering Systems

Early question answering systems are based on the closed corpora and closed domain approach, in which the QA system answers queries based on the questions present in a known domain and corpus. Examples are the QA systems such as BASEBALL [10] and Lunar [28]. These systems are specific to a domain and they hardly scale for the needs of WWW. On the other hand, the Morpheus system is built based on a more dynamic approach, where we make use of deep web sources and methods from prior successful searches to answer questions [9].

START[1] is a QA system that is built based on natural language annotation and parsing of the user queries. START answers queries by matching the annotated questions with candidate answers that are components of text and multi-media. Another method to question answering is a community based approach [9], where questions are answered by exploiting the responses to the questions posted by the users in a community. Yahoo! Answers [17] and Aardvark [15] are leading community based QA systems. Morpheus annotates user query terms by classes from an ontology with the help of *path finder*. This

[1] `http://start.csail.mit.edu`

annotated classes are essential in finding similarities between a newly encountered user query and the solved queries in the data store.

2.2 Ontology

An ontology formally models real world *concepts* and their relationships. A concept in an ontology gives us an abstract and simplified view of the real world [12] in a machine-readable and language-independent format. In addition, each ontological concept should have a single meaning in an ontological domain.

As mentioned in [19], one of the main reason for building an ontology is to share a common understanding of *structural* information among people and machines. For example, suppose we have many resources such as web pages, videos, and articles that describe the domain *vehicle dynamics*. If all of them share an inherent ontology for the terms they all use, the information extraction and aggregation can be done easily. In addition, an ontology enables the reuse of concepts [19]. For example, if one can build a comprehensive ontology about a specific domain (e.g. automotive), people from a different domain can easily reuse it.

In an ontology, a *concept* is formally described using the *class* construct . The relationships and attributes of an ontological concept are defined using *properties* and *property restrictions*. An ontology definition contains all of these: classes, properties, and restrictions. A class in an ontology can have multiple *super classes* and *sub-classes*. A sub-class of a class represents concepts that are more specific in the ontology domain, while super classes of a class represent more general concepts in the ontology domain. Finally, a *knowledge base* represents an ontology together with instances of its classes.

2.2.1 Ontology Building

Determining the *scope* and *domain* of an ontology is one of the challenging questions in ontology building. One way to model a domain or *realm* is to base it on a particular application [19]. In the Morpheus system, we focus on modeling user queries with in a particular realm [9]. Thus, we use the query realm as the ontology's *domain*. For example,

for user queries in an automotive domain, we create an automotive ontology. Moreover, we can extend the *scope* of an ontology to other domains easily if we follow an iterative ontology building process [19].

2.2.2 The Ontology Representations

As of 2004, the Web Ontology Language or OWL [2] is a standard for formally presenting an ontology. OWL has much stronger machine interpret-ability than RDF [3] . In this thesis, we follow the OWL standard to represent an ontology including classes, individuals, and properties.

2.3 Morpheus

If we search for an answer to a question in a typical search engine such as Google, Bing or Yahoo, it usually gives us relevant pages based on the key words in the query. We may need to follow several links or pages to reach a document providing a relevant answer. If we can store such search pathways to an answer for a given user query and reuse it for future searches it may speed up this process. Morpheus reuses prior web search pathways to yield an answer to a user query [9]. In fact, it provides tools to mark search trails and to follow path finders to their destinations.

Morpheus's QA strategy is based on exploiting deep (or hidden) web sources to solve questions, since many of deep web sources' quality information is coming from on-line data stores [18]. Web pages are just an interface to this information. Thus, in the Morpheus system, we explore these web pages to learn the characteristics of data, associated with each deep web locations.

Morpheus employs two types of users. One type, the *path finder*, enters queries in the Morpheus web interface and searches for an answer to the query using an instrumented web browser [9]. This tool tracks the pathways to the page where the path finder finds

[2] `http://www.w3schools.com/rdf/rdf_owl.asp`

[3] `http://www.w3schools.com/rdf/rdf_reference.asp`

the answer and other information required to revisit the path. The other type, the *path follower*, uses the Morpheus system much like a conventional search engine [9]. The path follower types a natural language question in a text box and gets a response from the system. The system uses previously answered queries and their paths to answer new questions.

To exploit the prior searches one should represent queries in an elegant way so that similar queries can be found efficiently and their stored search paths revisited to produce an answer. Morpheus uses a realm-based approach to represent the queries as described in the following sections.

2.3.1 The Morpheus Architecture

This section presents the core elements of the Morpheus system and its role in the *automated question answering* process. This architecture is originally published in [9].

Morpheus Ontology and Corpora:

Morpheus requires an ontology that contains classes of a particular realm. For example, the realm-ontology *Automotive* contains categories that are relevant to automobiles. In addition, each leaf class-node in the ontology is associated with a corpus of documents that are tagged to that class.

For realm ontology construction, Morpheus uses the Wikipedia categories (as ontological classes) and DBpedia ontology (as class taxonomy). Morpheus creates a realm ontology as follows: For a given realm, we first find a mapping category from the DBpedia ontology. Then we grab all the neighboring categories using a concept called a *Markov Blanket* [20] in *Bayesian Networks*. In fact, we grab the category's parents, its children, and its children's other parents using the DBpedia ontology properties *broader* and *narrower*. Once the blanket is ready, it grabs all parent categories of the categories in the blanket till it reaches root. Finally, we grab Wikipedia documents those are tagged

with the DBpedia categories and process the document terms[4] to construct corpus for each of the ontological leaf nodes [1, 9].

Associating path follower's query terms with classes:

From a class document corpus, we calculate the probability that a term belongs to a particular class. In fact, we determine the likelihood of a class given a term using *Bayes Rule* (Eq. 2–1). Here, the term-class and term-corpus probabilities are determined as relative frequencies [9]. The higher the probability, the more likely a term is referring to a class.

$$P(class|term) = \frac{P(term|class)P(class)}{P(term)} \tag{2–1}$$

Semi-Structured Query:

Morpheus represents user-query details in a semi structured format called a Semi-Structured Query or SSQ. It includes a user query in natural language format, relevant terms for the query, and place holders for the term *classes*. A term class represents a *topic* or *class* to which a term belongs. For example, the term *tire* can be classified into the *vehicle part* class. Morpheus assumes the term classes are from a consistent realm-based ontology, that is, one having a singly rooted heterarchy whose subclass/superclass relations have meaningful semantic interpretations.

Query Resolution Recorder:

The *Query Resolution Recorder* or QRR is an instrumented web browser used to track the search pathways to an answer for a *path finder's* query. It also helps a path finder to identify ontological classes associated with search terms [8, 9]. Morpheus captures all these information in the format of an SSQ. For example, suppose, a path finder tries to answer the query " What is the tire size for a 1997 Toyota Camry V6? " using Morpheus. With the help of the QRR, a path finder may choose relevant query terms and assign

[4] *word*, *n-gram*, and *term* are used interchangeably in this thesis

categories to them (see table 2-1). Moreover, using QRR path finder is able to log the search pathways to an answer (e.g. P215/60R16) to this query and select a query realm (e.g. automotive).

Table 2-1. Example semi-structured query

Terms	1997	Toyota	Camry	V6	tire size
Input	year	manufacturer	model	engine	
Output					tire_size

Query Response Method:

The Morpheus QA process is formalized by the Query Resolution Method or QRM data structure [9]. A QRM is usually constructed by a path finder with the help of the QRR. Each QRM contains an ontological realm, an SSQ, and information to support the query answering process. QRMs are associated with an ontology that has a particular realm, i.e., an ontological realm. The associated SSQ contains a user query, selected terms and class associations, and output categories for a given query. In addition, the QRM contains information required to visit each page needed to answer the query as well as the order of visitation. For each dynamic page, it stores the list of inputs to the URL such as form inputs and the possible reference outputs (highlighted portions of the web page).

NLP Engine:

The path follower's query is processed in a different approach. The NLP engine parses and annotates queries in order to record important terms. Based on the query terms, Morpheus assigns the most probable query-realm from realm-specific copora. Once the realm is assigned, we use the corresponding realm-ontology and associated corpora to assign classes to the terms as mentioned above [9]. Subsequently, the system creates an SSQ and this new SSQ is used to match with existing QRM SSQs in the store.

QRM Ranking Mechanism:

To answer a path follower's query that is in SSQ format (a *candidate SSQ*), Morpheus finds similar SSQs that belong to QRMs in the store (*qualified SSQs*). The similarity between a *candidate SSQ* and a *qualified SSQ* is calculated by aggregating the similarity

measures of their assigned *classes* [9]. Class similarity is based on the heterarchical structure of classes in an ontology. Chapter 4 describes the Morpheus *class similarity measure* in detail.

Query Resolution Execution:

Usually, answering a question using the deep web requires one to navigate through a sequence of one or more web pages. Many of the accesses involve clicks through web forms or links to resulting pages. Each QRM contains the necessary information required to re-run this procedure. Once we have found relevant QRMs, through QRM ranking for a given path follower query, Morpheus gets answers by re-running the stored pathways in QRM. The Morpheus Query Executer performs this job by simulating the human button clicks to follow links, form submission, and HTML page answer highlighting. This also has the routines necessary to generalize the QRM query paths to tackle problems associated with the dynamic generation of the deep web pages [9].

2.3.2 Discussion

The backbone of the Morpheus system is a realm ontology and the corpora associated with the ontology's categories. Since the Morpheus ontology structure is tightly coupled to the DBpedia ontology, let us consider the DBpedia ontology. DBpedia is created from an automated process that grabs info-box information and the class heterarchy from from Wikipedia pages. Thus, the correctness of the DBpedia ontology greatly depends on the Wikipedia class structure.

As mentioned in YAGO [26], to use an ontology for type information its classes must be arranged in a taxonomy. Even though DBpedia categories are arranged in a heterarchy, it cannot be used as a robust ontology. For example, "Effects of the automobile on societies" is in the class named automobiles, but "Effects of the automobile on societies" represents a result of automobiles and not information about automobiles.

Another draw back of blindly using the Wikipedia categories and their associated Wikipedia pages for corpus construction is that categories that are assigned to Wikipedia

pages do not follow any standard. Similarly, the class assignment for a Wikipedia page is not mandatory, so many pages do not have any assigned class.

As discussed above, the DBpedia ontology and Wikipedia page categorizations are not adequate to Morpheus's needs, even though these too contains huge amounts of information. According to alexa.com Wikipedia is the 9th most visited website as of May 2010 and it covers almost all fields. So, if we can make a realm-based ontology from the Wikipedia this may help in the process of question answering.

2.4 Ontology Generation Systems

This section talks about existing ontology generators and their applicability in building a Morpheus realm ontology from Wikipedia pages.

2.4.1 DBpedia Ontology

The DBpedia[5] project is based on extracting semantic information from the Wikipedia and making it available on the Web [1]. Wikipedia semantics includes info-box templates, categorization information, images, Geo-coordinates, links to external Web pages, disambiguation pages, and redirects between pages in Wiki markup form [1, 5, 13]. The DBpedia maintainers process this structured information and store it in RDF triple format.

The extraction procedure is as follows: Wikipedia categories are represented using skos:concepts and category relations are represented using skos:broader [1, 5]. In fact, DBpedia does not define any new relation between the Wikipedia categories. The Wikipedia categories are extracted directly from Wikipedia pages and there are no quality check on the resultant ontology. Similarly, DBpedia uses the info-box attributes and values as ontological relation domains and their ranges.

Since the guidelines for info-box templates are not strictly followed by the Wikipedia editors, info-box attributes are in a wide range of different formats and units of measurement

[5] `http://dbpedia.org`

[5]. DBpedia has taken care of these issues using two approaches (1) *A generic approach* for mass extraction: All the info-boxes are processed into triples with the subject being the page URI; predicate as the attribute name, and object as the attribute's value. Even though several heuristics can be applied, the information quality is not guaranteed [1, 5] (2) *A mapping-based approach* for maintaining information quality: By manually mapping info-box templates into an ontology, this method solves the issues of synonymous attribute names and templates.

2.4.2 WordNet Ontology

WordNet[6] is a huge manually built lexicon of English. WordNet contains nouns, verbs, adjectives, and adverbs that are grouped into synonyms' sets or *synsets*. The semantic and lexical relations such as *hypernym* and *hyponym* connect the WordNet synsets. These relations form a heterarchy of synsets and are very useful in the ontology building process. One of the big advantages of using the man-made WordNet heterarchy for ontology building is that it has a consistent taxonomy of *concepts*, which is a requirement in the ontology definition[2.2]. However, the cost of keeping them up to date is very high. In addition, if you consider the entire WWW WordNet represents a small amount of data. To overcome these issues, this thesis proposes a set of tools that assist automatic ontology building process, and suggests their extensions to include other domains or realms.

2.4.3 YAGO

YAGO is a partly automatically constructed ontology from Wikipedia and WordNet [26]. The YAGO ontology uses an automated process to extract information from Wikipedia pages, info-boxes, and categories and to combine this information with the WordNet synset heterarchy. Since Wikipedia contains more individuals in the form of Wikipedia pages than in the man-made WordNet ontology, the Wikipedia page titles

[6] `http://wordnet.princeton.edu`

are used as the YAGO ontology individuals. YAGO concentrates mainly on the fields of people, cities, events, and movies [26].

In addition, YAGO uses Wikipedia categories as its ontology classes. The Wikipedia categories are classified into four types such as conceptual, relational, administrative, and thematic by parsing the Wikipedia category names. Only the conceptual and relational categories are used for the ontology building process [26]. The ontology's heterarchy is built using the *hypernym* and *hyponym* relations of the WordNet synsets. YAGO uses only the nouns from WordNet and ignores the WordNet verbs and adjectives. The connection between a WordNet synset and a Wikipedia category is achieved by parsing the category names and matching the parsed category components with the WordNet synsets [26]. Those Wikipedia categories having no WordNet match are ignored in the YAGO ontology.

Additional ontology relationships and individuals are generated by parsing Wikipedia page info-box attributes and values. This information extraction is done with the help of attribute maps of info-box templates and regular expressions [26]. Commonly used info-box templates are also used as classes in the ontology (e.g. outdoors). One drawback of this approach is that info-box assignment is not a mandatory task in creating a new Wikipedia page. In addition, many Wikipedia pages do not have any categories assigned to them. Therefore, using info-box-based information extraction is not feasible for these Wikipedia pages.

In addition, YAGO type extraction is mainly based on the assumption that each Wikipedia page has at least one tagged category, and the assigned Wikipedia categories are relevant to that Wikipedia page. As explained in section 2.4.1, the Wikipedia page category assignments are subjectively assigned by human editors. Some Wiki pages could be left unassigned i.e., they have no category unintentionally. In addition, we cannot completely rely on the relevance of these assigned Wikipedia page categories. This thesis investigates methods for automatic web page categorization and recommendation, using

a machine learning technique, topic modeling. The rest of this chapter describes existing document modeling and classification methods.

2.5 Topic Models and Document Classification

There have been many studies conducted in modeling text documents [25] in the Information Retrieval field. Probabilistic Latent Semantic Indexing (pLSI), a model introduced by Hoffmann [14], uses a probabilistic generative process to model the documents in a text corpus. In this model, we assume that every word in a corpus document is sampled from a mixture model. Similarly, components of the mixture model are multinomial random variables that represent topics in the corpus [14]. Thus, a corpus document in the corpus can have words from multiple topics and the mixture components' mixing proportions represent it [21]. However, pLSI draws each word from one topic. Even though it has many advantages, the number of model parameters increases linearly with the number of documents in the corpus [6]. In addition, it has an inappropriate generative semantics, and the model does not have a natural way to assign probabilities to unseen data [27].

Latent Dirichlet Allocation (LDA) is a probabilistic, graphic model for representing latent topics or hidden variables of the documents in a text corpus [6]. Due to its fully generative semantics, even at the level of documents, LDA could address the drawbacks of the pLSI model. In this hierarchical Bayesian model, LDA represents a document in the corpus by distributions over topics. For example, for Wikipedia pages, the latent topics reflect the thematic structure of the pages. In addition, a topic itself is a distribution over all terms in the corpus. Thus, LDA can find out relevant topic proportions in a document using posterior inference [6]. Additionally, given a text document, using LDA we could tag related documents by matching the similarity over the estimated topic proportions. This thesis mainly exploits the classification capability of the LDA model [27].

2.6 Summary

This chapter gave an introduction to question answering systems and a brief overview of the Morpheus system architecture. It also discussed the use of an ontology and corresponding corpora in the Morpheus QA process, and conventional ontology generation methods. Extracting topics from text and categorizing documents are essential in building an ontology and the associated class-corpora. This chapter gave an introduction to document modeling systems and explained how LDA provides a better model than other strategies. The next chapter discusses the applications of LDA for efficient data analysis of large collections of web documents, and how this help us building an ontology. This is followed by a query ranking algorithm that is built on top of the ontological heterarchy.

CHAPTER 3
DOCUMENT MODELING

The realm ontology is one of the pillars of Morpheus's query matching algorithm. This chapter describes the tools that helps to build a realm ontology in a semi-automated fashion. First, I discuss Latent Dirichlet Allocation (LDA), a probabilistic topic model used in this thesis for modeling and classifying Wikipedia pages. Second, I present algorithms for classifying the Wikipedia pages, and how we can use the categorized web pages for a realm ontology and corpora. In addition, the results section explains the observations from the LDA-based document classifier and their interpretations. This chapter concludes by describing the global impacts of these approaches in WWW page categorization and ontology generators.

3.1 Latent Dirichlet Allocation

As described in section 2.5 LDA has some promising advantages over the probabilistic LSI (pLSI) model in document modeling due to its fully generative semantics [6, 27]. LDA assumes that the mixture proportions of topics, for documents in a collection, are drawn from a Dirichlet prior distribution. However, in the pLSI model the mixture proportions of topics are conditioned on the training documents. Therefore, in the pLSI model the number of model parameters grows linearly with the number of documents in a training set. LDA's generative process employs the following concepts (Notation and terminology adopted from [6, 27]):

- w represents a word or a *term* in a corpus vocabulary, and V represents the size of a corpus vocabulary.
- $\mathbf{w} = (w_1, w_2, ..., w_N)$ is a document in a text corpus, where w_i is the i^{th} word in the document, and N_d stands for the total number of terms in a document that is indexed by d. LDA assumes that a document is a *bag-of-words*.
- $C = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_D\}$ is a text corpus, where D is for the number of documents in a corpus.

The generative process of LDA is the following [6, 27]:

- Choose each topic $\vec{\beta_k}$ from a Dirichlet distribution $Dir_V(\eta)$ over the vocabulary of terms, in other words $\vec{\beta_k}$ represents topic-mixture over the entire corpus of terms. Here, we assume *Dirichlet smoothing*[6] on the multinomial parameter β_k.
- For each document d choose $\vec{\theta_d}$, a vector of topic proportions, from a Dirichlet distribution $Dir(\alpha)$ with hyper parameter α
- For each of the N_d words w_n in a document $\mathbf{w}_d$
  - Choose a topic $z_{d,n}$ from a multinomial distribution $Mult(\vec{\theta_d})$ with parameter $\vec{\theta_d}$
  - Then, choose w_n from a multinomial distribution $p(w_n|z_{d,n}, \vec{\beta})$ conditioned on $z_{d,n}$ and $\vec{\beta}$

Thus, the joint probability distribution of a topic mixture $\vec{\theta_d}$, a set of N topics $\mathbf{z}$, and a set of N words w_n in a document $\mathbf{w}_d$ is:

$$p(\vec{\theta_d}, \mathbf{z}, \mathbf{w}) = p(\vec{\theta_d}|\alpha)\, p(\vec{\beta}|\eta) \prod_{n=1}^{N} p(z_{d,n}|\vec{\theta_d})\, p(w_n|z_{d,n}, \vec{\beta}) \tag{3–1}$$

where $p(\vec{\theta_d}|\alpha)$ and $p(\vec{\beta}|\eta)$ are Dirichlet distributions. Then, the marginal distribution of a document d is obtained by summing over z and $\vec{\theta_d}$ as:

$$p(d|\alpha, \eta) = \int p(\vec{\theta_d}|\alpha)\, p(\vec{\beta}|\eta) \prod_{n=1}^{N} \sum_{z_{d,n}} p(z_{d,n}|\vec{\theta_d})\, p(w_n|z_{d,n}, \vec{\beta})\, d\vec{\theta_d} \tag{3–2}$$

Finally, assuming *exchange-ability* over the text documents in a corpus, the probability of a corpus is given by multiplying the probabilities of single documents[6]. Figure 3-1 shows the LDA graphical model.

From this LDA model, we can infer the hidden parameters such as per-word topic assignment z_n, per-document topic proportions θ_d, per-corpus topic distributions β_k. However, the posterior probability formed by the equation 3–2 is intractable to compute[6]. Therefore, we usually pursue different approximation techniques such as *Variational Inference* [6] and *Markov Chain Monte Carlo* [11] to estimate the hidden parameters. Section 3.3 describes more on the Wikipedia page modeling using LDA.

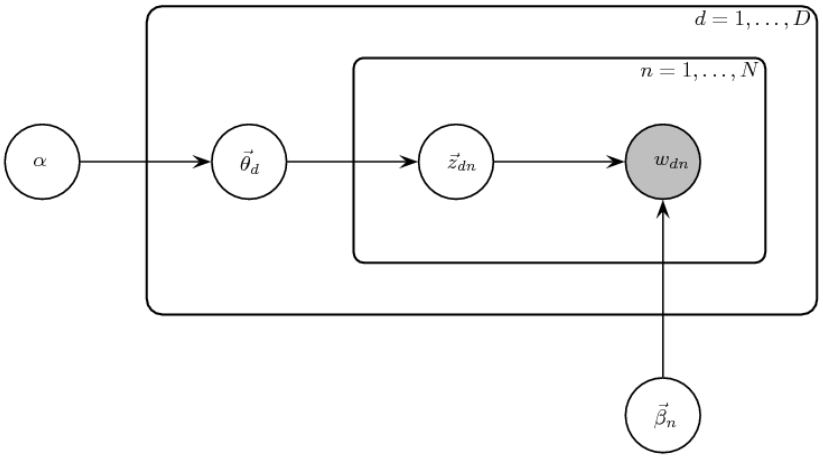

Figure 3-1. The graphical model of LDA: The nodes represent random variables, shaded nodes stand for the observed random variables, and rectangles are for replicated structures.

3.2 Feature Extraction

For building a machine learning model for text categorization we need a training set and test set. This thesis uses pre-processed text from the Wikipedia pages to build the training set and test set. The Wikipedia pages for the data set building are identified with the help of the Wikipedia category heterarchy and associated web pages. In the pre-processing step, the Wikipedia page text is tokenized into *terms* or word tokens using Natural Language Processing Toolkit (NLTK) [1] and regular expressions. Moreover, these terms are processed further for standardizing terms and removing term redundancy. The following sections describe these techniques in detail.

3.2.1 Lemmatization

Lemmatization is a standardization technique that changes different inflected forms of a term into a common a form *Lemma*, so that we can use Lemma to represent a set of terms (e.g. appeared → appear, women → woman). In text processing, *lemmatization*

[1] `http://www.nltk.org`

helps us to reduce the dimensionality of words in a corpus vocabulary. This thesis uses the WordNet based lemmatizer from NLTK.

3.2.2 Stemming

Stemming is a process to get *stems* from terms by removing unnecessary grammatical markings (e.g. walking → walk). There are many stemming algorithms exist - Porter, Lancaster, etc. This thesis uses the Porter stemmer for the words that cannot be lemmatized using the WordNet lemmatizer[3].

3.2.3 Feature Selection or Reduction

Feature selection is a process that removes uninformative terms or noise from the extracted text. By reducing the dimensionality of feature-vectors that are used to represent the text documents and corpus vocabulary, feature selection speed-ups the language modeling process. This thesis uses a filter based approach by removing the *stop-words* of English from the extracted tokens. The stop-words are the terms that are common to any document or the terms that are specific to a language or grammar. Another approach to feature selection is based on the classifier or clustering algorithms output. The end of this chapter describes a feature selection method based on the LDA model output.

3.2.4 Representing Documents

This thesis uses a sparse vector space model to represent the text documents in a corpus. Features are the frequencies of terms or words in a document. Each document in a corpus is represented by a sparse vector[2] as:

$$\mathbf{W}_d = \{\, id_i : count \,\}_{id_i \,\epsilon\, \mathbf{V}} \tag{3–3}$$

where $\mathbf{V}$ represents the corpus vocabulary, id_i is the unique vocabulary id, and $count$ represents the number of occurrences of term in the document $\mathbf{W}_d$.

[2] `http://www.cs.princeton.edu/ blei/lda-c`

3.3 Document Modeling using LDA

This section describes LDA as a tool to topic extraction and feature selection from text. Firstly, we go through the topic extraction from web documents and the limitations of using the extracted LDA topics for the ontology learning process. Secondly, LDA is used as a tool for dimensionality reduction[6] and feature selection[30]. Lastly, we go through a classifier that is built based on the LDA output, for the unseen documents in a corpus.

The parameter estimation of LDA is performed using Gibbs sampling[11] on the preprocessed documents from Wikipedia. This thesis uses the $\mathcal{R}$ package **lda**[3] for the parameter estimation process. The end results of this estimation are a document-topic matrix ϕ and a word-topic matrix β. ϕ is a K X D matrix where each entry represents the frequency of words in each document that were assigned to each topic. The variable K, the total number of topics in a corpus, is an input to the model. β is a K X V matrix where each entry is the frequency of a word was assigned to a topic. The following sections explains the experiments conducted using the LDA model.

3.3.1 LDA for Topic Extraction

For this experimental study, the Wikipedia pages from the *Whales* and *Tires* domains are identified using the DBpedia SPARQL[4] interface. For the Whales domain, the Wikipedia categories *Whale_Products, Whaling, Baleen_Whale, Toothed_Whale, and Killer_Whale* (119 web pages) are selected and for Tires domain the Wikipedia category *Tires* and its sub-categories (111 web pages) are selected. After pre-processing [section 3.2], the document terms are used for the LDA model training. Table 3-1 shows the three topics that are extracted from the whales-tires documents. The topics' words are listed in the non-increasing order of probabilities $p(topic \mid term)$. Table 3-1 shows that there

[3] `http://cran.r-project.org/web/packages/lda`

[4] `http://dbpedia.org/sparql`

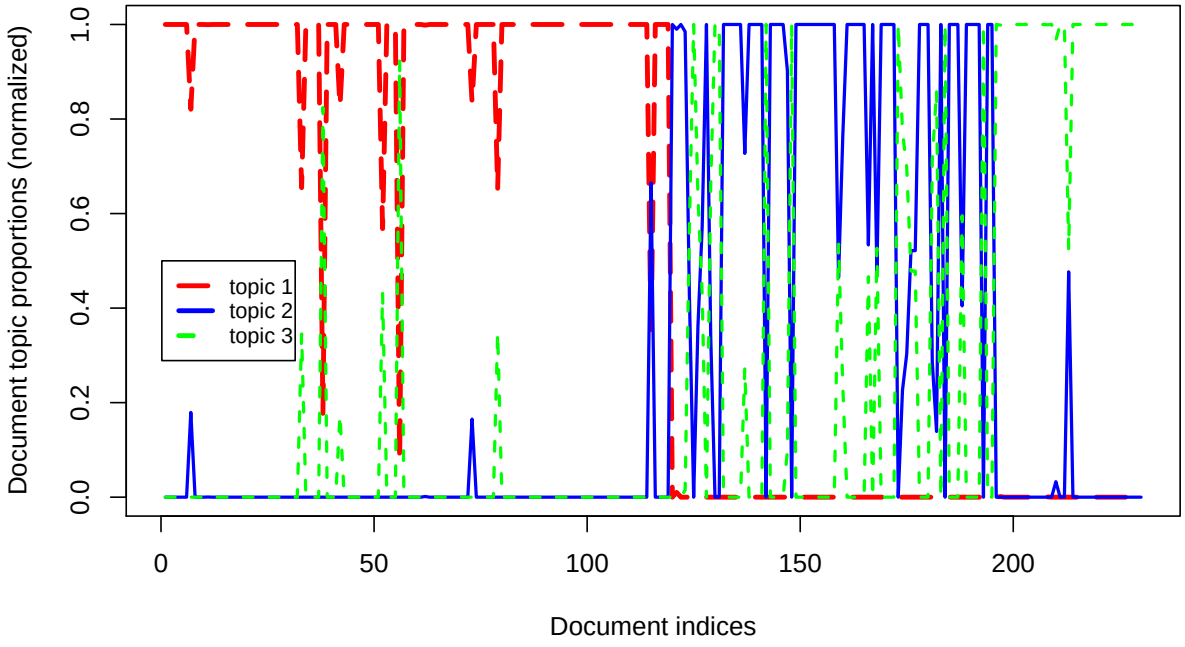

Figure 3-2. LDA topic proportions the training documents from the whales (first 119) and tires' (last 111) domains. The topics from the LDA model, with three topics, are represented by different colored lines.

are two topics (whales and tires) that are dominant in the corpus documents. Figure 3-2 represents the corresponding normalized topic proportions of the training documents.

Table 3-1. Estimated topic-words using LDA with three topics

Number	Topic 1	Topic 2	Topic 3
1	whale	tire	company
2	dolphin	wheel	tyre
3	species	vehicle	new
4	sea	tread	tire
5	ship	use	bridgeston
6	killer	pressure	plant
7	iwc	wear	rubber
8	orca	system	goodyear
9	population	valve	firestone
10	animal	car	dunlop

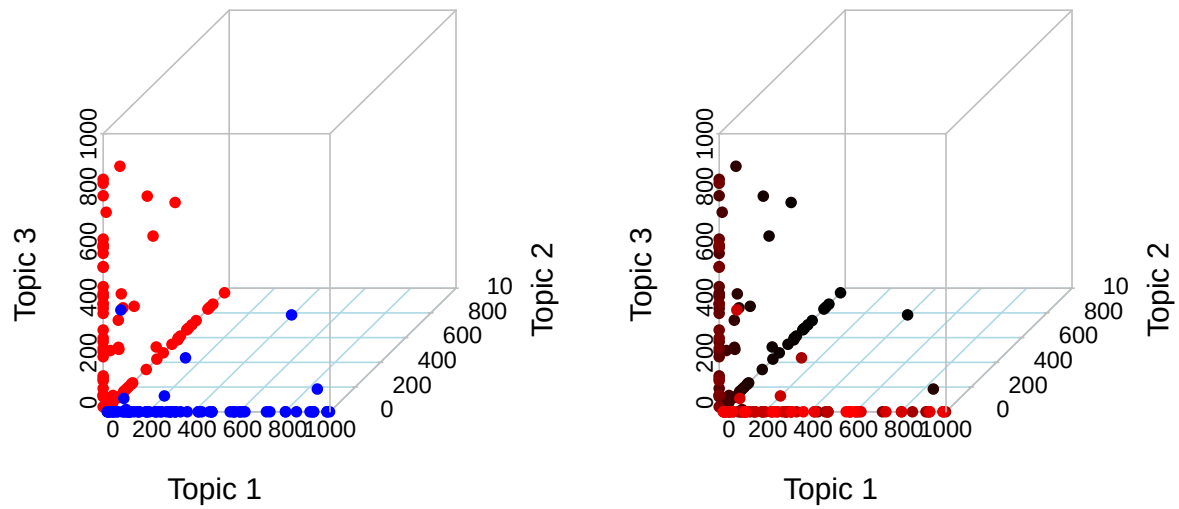

Figure 3-3. The topic proportions of the whales (first 119) and tires' (last 111) domains from the LDA model with three topics. Left plot represents documents from both classes in the topic space, red for whales and blue for tires. Right plot represents the document topic proportions in 3-D for the two classes

Additional experimental results can be seen from (figure 3-3, figure 3-4, table 3-2). From these experiments we can infer that, if the documents are completely different domains (e.g. whales and tires), LDA finds the topics and document-topic-mixture that can be used in classifying the corpus documents. However, if the documents are from similar domains (e.g. the sub domains of whales), LDA fails to capture distinguishable topics (figure 3-2). In addition, LDA defines topics over all the terms in a vocabulary. Therefore, topics extracted from the LDA estimation process are sets of words, not names of categories similar to the words used to tag Wikipedia pages. The common reason for the above two issues is the way LDA is designed. It is a probabilistic model based on the frequency of terms in the corpus documents and not based on the terms' meanings. Even though the documents are tagged to different domains, if the documents have the same terms, LDA will capture the similarity of their topics. Table 3-2 also shows the importance

Table 3-2. Estimated topic-words using LDA with two topics

Number	Topic 1	Topic 2
1	whale	tire
2	dolphin	tyre
3	species	wheel
4	sea	rubber
5	ship	vehicle
6	killer	tread
7	iwc	car
8	orca	pressure
9	population	wear
10	animal	system

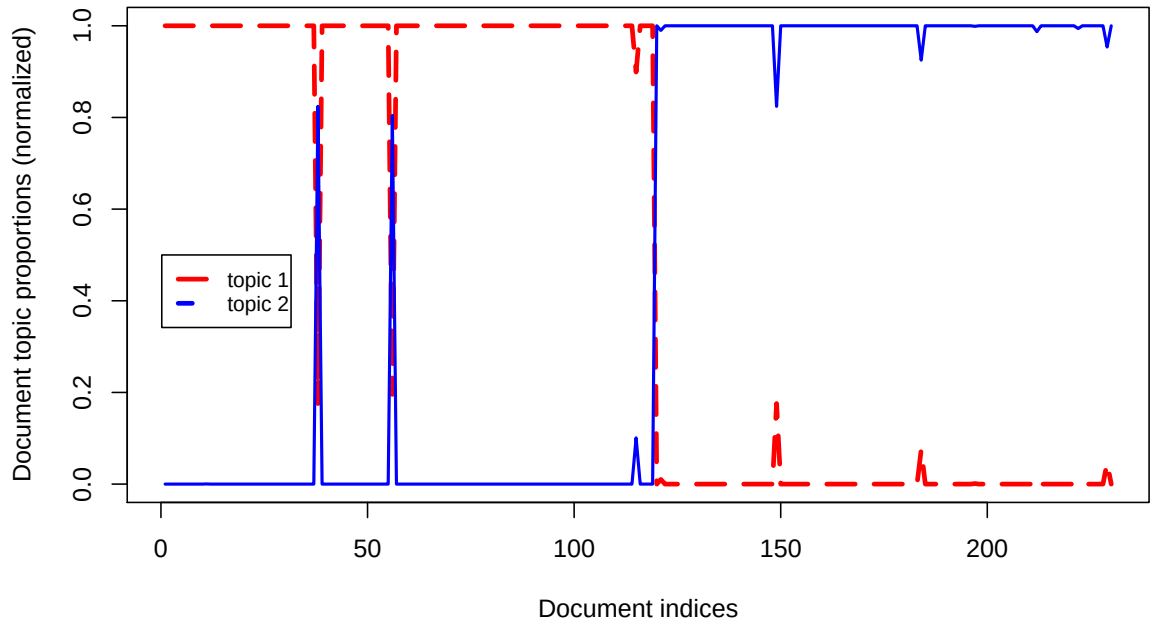

Figure 3-4. The two topic LDA's topic proportions for the whales (first 119) and tires (last 111) documents

of lemmatizing and standardizing the terms in the LDA estimation. We see that *tire* and *tyre* both appear in *Topic* set 2, but they have the same meaning.

3.3.2 LDA for Dimensionality Reduction

LDA can be used for dimensionality reduction of the corpus documents in two ways. One method is to represent documents by a document-topic-mixture vector $\vec{\theta}_j$

from the LDA estimation instead of the document word frequency vector[6]. Thus, we can reduce the dimensionality of a document from vocabulary size to the number of topics in a corpus. If LDA can find the *actual* number of topics in a corpus, documents will be linearly separable using the document-topic-mixture vectors. *Actual* denotes the topics (e.g. whales and tires) that are used to manually tag the documents in the corpus. For example, figure 3-4 shows that an individual topic mixture proportion is enough for distinguishing the documents from the whales and tires domain. In fact, this document-topic matrix can be used for clustering or classifying the corpus documents[6].

Another approach to dimensionality reduction of the document terms is based on the term-entropy values calculated on the term-topic matrix β [30]. Each element of β represents the conditional probability of a term v given a topic k, $p(term_v|topic_k)$. Our aim is to find best discriminative terms in the corpus. Shannon entropy is a measure to calculate uncertainty of a system. So this fits well to find useful terms of classification or regression. Thus, the entropy is calculated for a term $term_v$ by [30]:

$$H(term) = -\sum_{k=1}^{K} p(term_v|topic_k) ln\, p(term_v|topic_k) \tag{3–4}$$

Figure 3-5 shows the term-entropy values calculated on the β matrix for the whales and tires documents. If the entropy values are high, that means that term is common among the corpus topics. So we can rank the terms in the non-decreasing order of entropy and discard the corpus terms by applying a threshold. The rest of the terms and its frequencies can be used are the new features for a document. Figure 3-6 shows the entire feature reduction process based on entropy [30].

Experiments conducted on the document from *whales* and *tires* domains shows that this feature selection process highly depended on the number of topics assigned to the LDA estimator. Table 3-3 shows the number of terms selected on the whales and tires documents when a common threshold (quantile 40%) is applied on term-entropies.

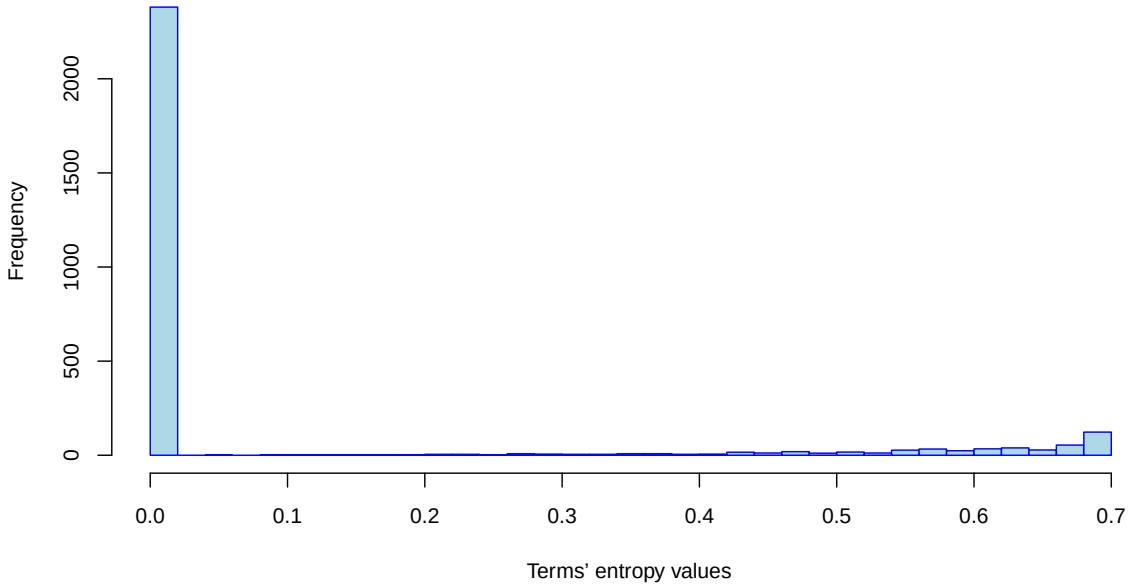

Figure 3-5. Histogram of term entropy values over the LDA β matrix: calculated on the data set whales and tires

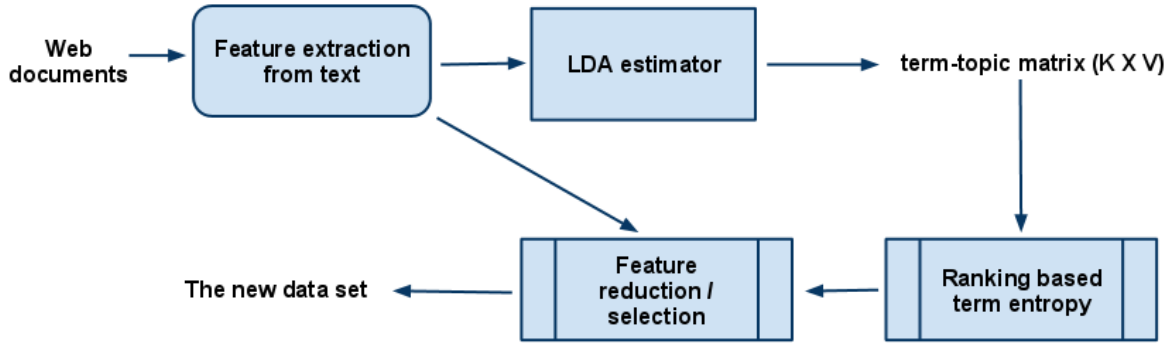

Figure 3-6. LDA based feature reduction

3.3.3 Document Classifier from LDA outputs

As discussed in section 3.3.1, we cannot directly use the estimated topics from LDA as the topic or class of documents for the ontology building, because the estimated topics are sets of words or a distribution over the vocabulary words. One way to tackle this problem is to build a document classifier using document-topic-mixture vectors from the

Table 3-3. LDA based feature reduction with different number of topics

	2 topics	3 topics	5 topics	10 topics	15 topics
Selected # of terms	2408	1664	899	246	104

LDA model's output. The following sections describes two different approaches to use the document-topic-mixtures in classifying the corpus documents and their results.

Document topic proportions' distance:

A naive approach is to find the centroid or mean $\hat{\vec{\theta}}$ of the tagged documents topic-mixtures $\vec{\theta}_d$ and classify the unseen documents into a particular class by using the minimal topic-mixture distance to the class mean. Two distances that can be used for this approach, are the *Cosine* and *Hellinger* [23] distances:

$$Cosine(\vec{\theta}_d, \hat{\vec{\theta}}) = \frac{\vec{\theta}_d \,.\, \hat{\vec{\theta}}}{\left\|\vec{\theta}_d\right\| \left\|\hat{\vec{\theta}}\right\|} \tag{3–5}$$

$$Hellinger(\vec{\theta}_d, \hat{\vec{\theta}}) = \sum_{k=1}^{K} (\sqrt{\vec{\theta}_d} - \sqrt{\hat{\vec{\theta}}})^2 \tag{3–6}$$

Figure 3-7 shows the corresponding distance plots for the whales (first 40 documents) and tires (last 37 documents) documents. In this experiment the LDA estimation, with 2 topics, is performed on all the documents in the data set, which is extracted from the Wikipedia pages in the whales and tires domain (230 documents). Then, the whale documents' mean-topic-proportion is calculated on the train set, $2/3^{rd}$ of the LDA document topic proportions matrix θ. Subsequently, the cosine and Hellinger distances are calculated on the test set, $1/3^{rd}$ of the θ matrix. Low Hellinger distance or high cosine distance values indicate the corresponding θ values are nearer to the class mean.

Moreover, we can build classifiers based on this approach by applying a threshold on the distances to a class document topic proportions' mean. For example, to build a classifier using Hellinger distance and whales-mean, we set the threshold as 0.5. Afterwards, we assign all the points below the threshold to the whales class and vice versa. Figure 3-8 shows the classification accuracy of the classifiers built based on this

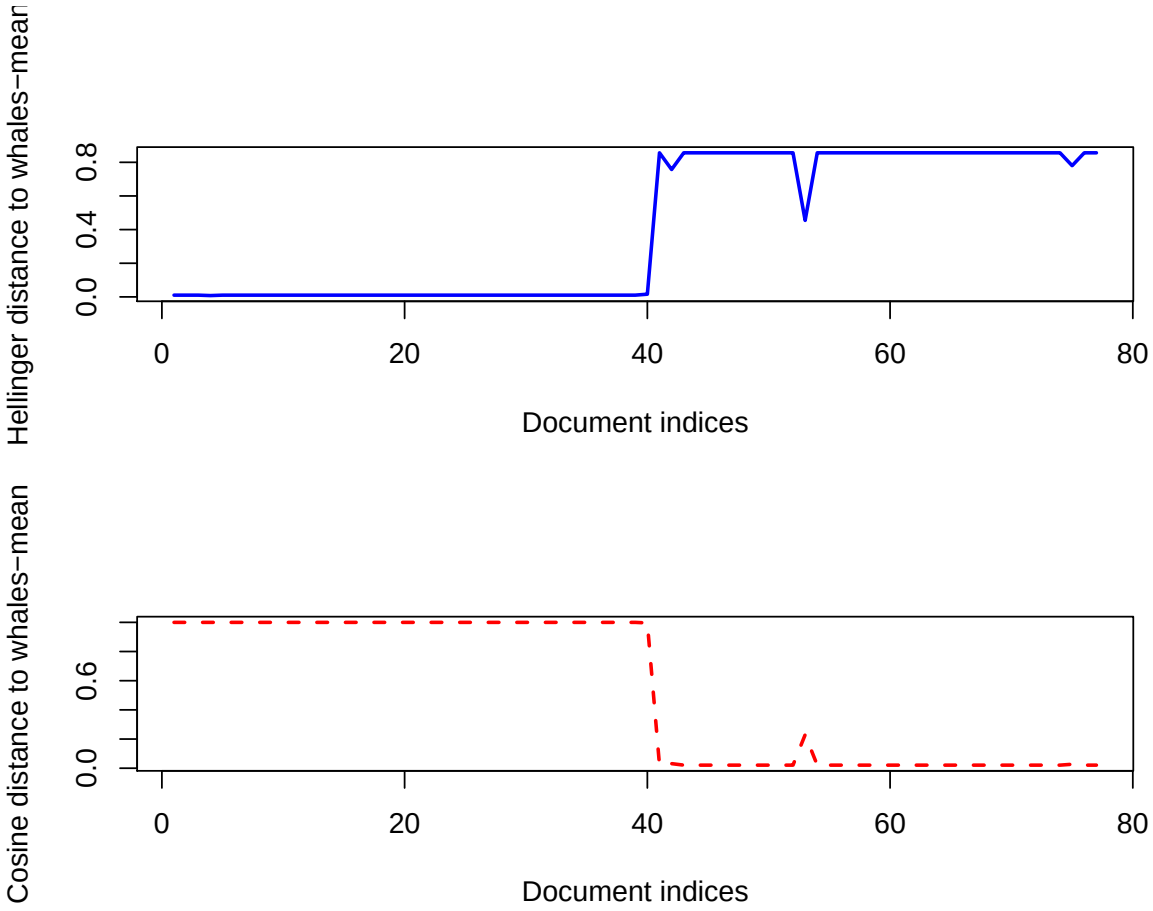

Figure 3-7. Hellinger (top) and cosine (bottom) distances of the document topic mixtures to whales' document topic mixture mean.

approach and with varying number of topics. From this, we can infer that when the number of topics vary from the desired number of topics (for this corpus the desired value is 2) in the corpus, the accuracy of the classifiers deteriorate. The bottom line is that one should find the accurate number of corpus topics for an LDA model before applying the LDA document topic mixtures to any classification purpose. Accuracy values can also be used for selecting the number of topics in an LDA model.

Maximum margin classifier

In this experiment a maximum margin classifier, a support vector machine (SVM) classifier (SVC), is built on the document topic mixture proportions from the LDA model. The training set is built on $2/3^{rd}$ of the LDA document topic proportions matrix θ, both

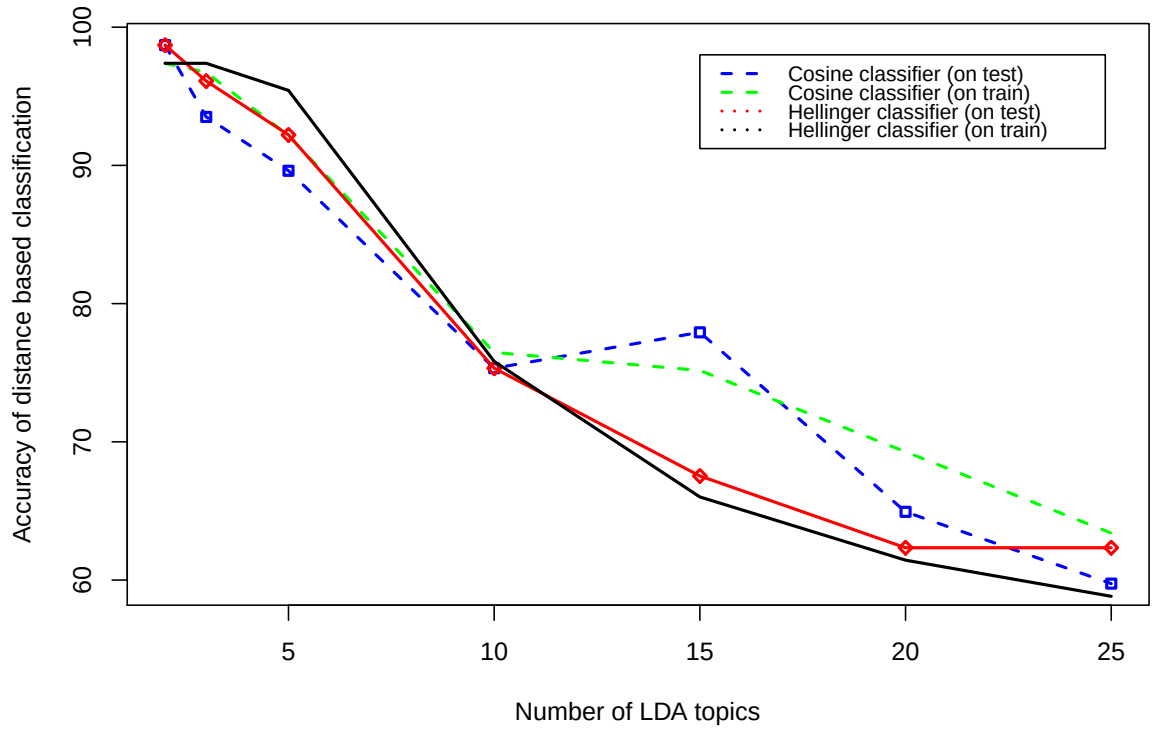

Figure 3-8. Classification accuracy of the classifiers build based on Hellinger and cosine distances with varying number of topics. The lines with bullets represents the accuracy values on test set

classes, whales and tires, are equally distributed. The test set is from the rest of the document topic proportions. This thesis uses the LIBSVM [7] package to train the SVM classifier. Figure 3-9 shows the flow diagram of this experiment. Figure 3-10 shows the accuracy of the SVM models on the test set with 10-fold cross validation [4], varying number of topics applied to the LDA model. From this, we can see that SVC performs about as well as the distance based classifiers [figure 3-8], when the LDA model is trained with desired number of topics. Moreover, the SVC models outperform the distance based classifiers, when the number of topics assigned to the LDA model is incremented.

Figure 3-12 shows the distance measures calculated to whales-mean on the same training set. It also shows the distance of support vectors (marked as S) that were found

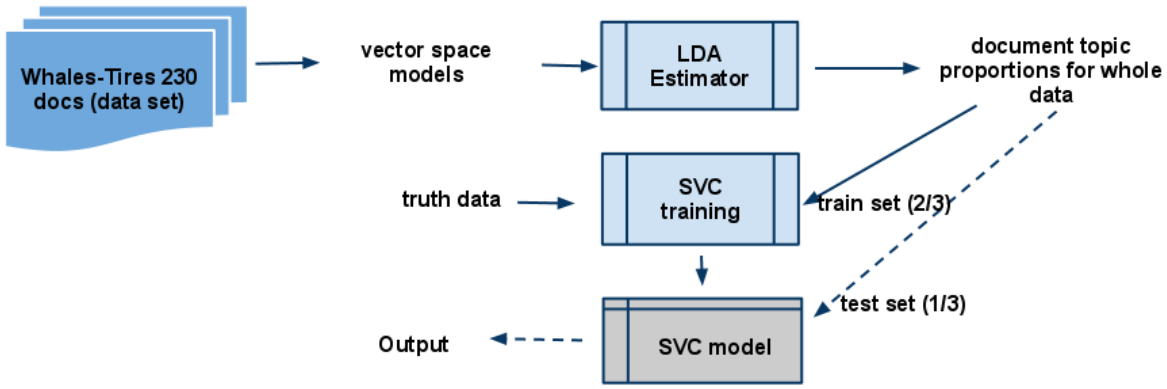

Figure 3-9. SVM classification model using the LDA document proportions

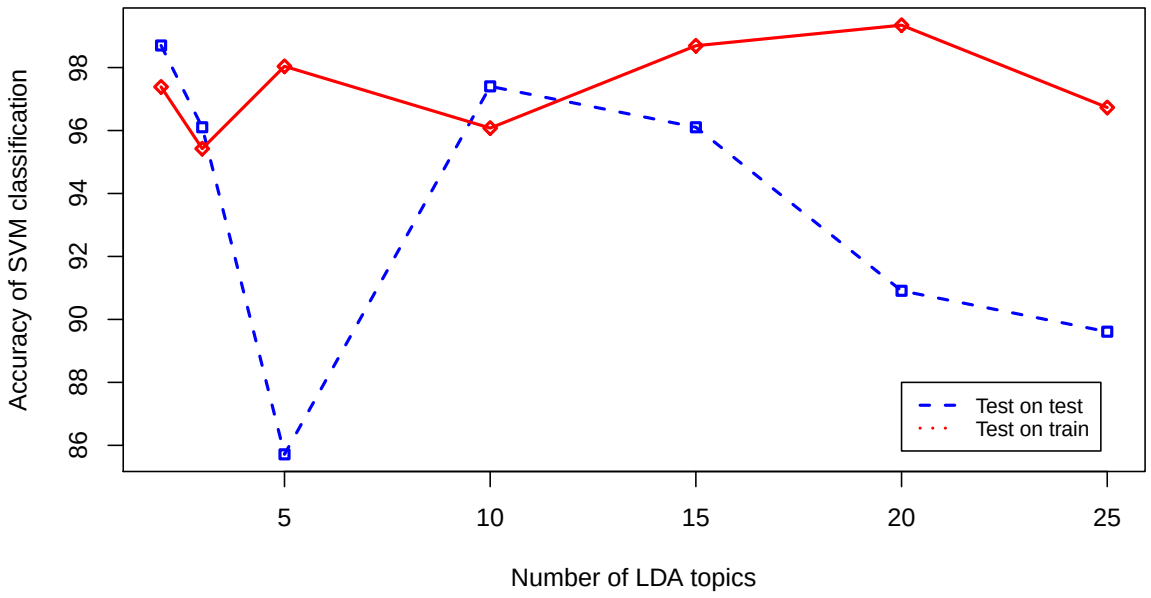

Figure 3-10. SVM classification accuracy of the whales-tires document topic mixtures on the varying number of topics

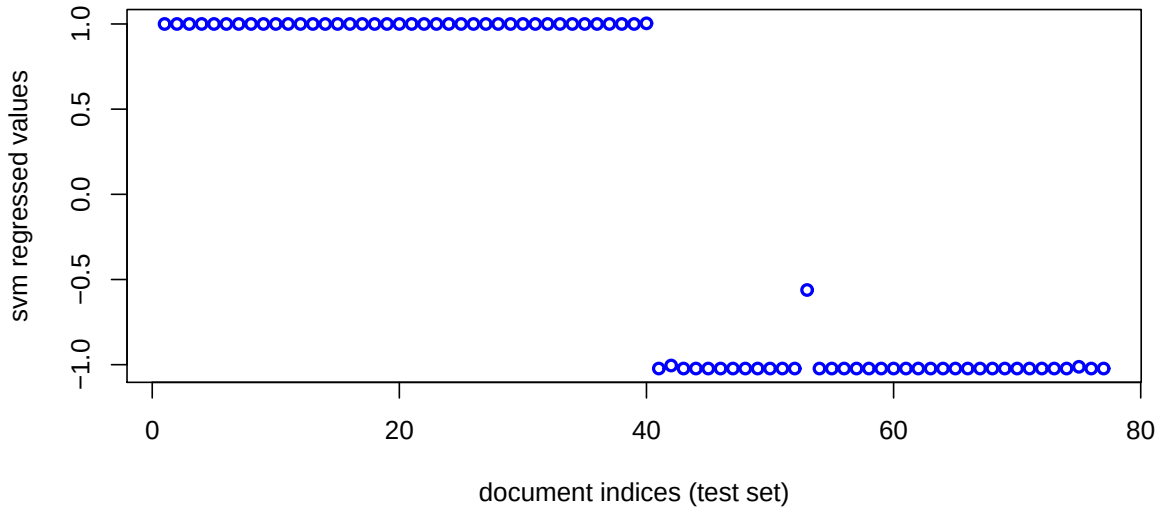

Figure 3-11. SVC regressed values of the whales-tires document topic mixtures with two topics.

in the SVC model built on the same training set to the whales-mean. This figure gives us an intuitive idea of the support vector selection from the training set. Figure 3-11 shows the SVM regressed values of the test set built from the whales-tires document topic mixtures with two topics. The tire document with its regression value above -1 (represents the tires class) is an outlier document. Even though, this document is tagged as tires category in Wikipedia, its terms are belong to whales class. We only need to store the SVC model, which is built on a huge corpus of documents, for future classification. This saves the storage space of a classification model.

3.3.4 Inferring Document Topic Proportions from the Learned LDA Model

Topic models are useful in analyzing large collection of documents and representing high-dimensional documents in low-dimensional form. However, fitting an LDA model from documents is computationally expensive, since it uses different approximation techniques such as Gibbs sampling or Variational methods. We can tackle this issue by inferring document-topic-mixture proportions for the unseen documents from the learned

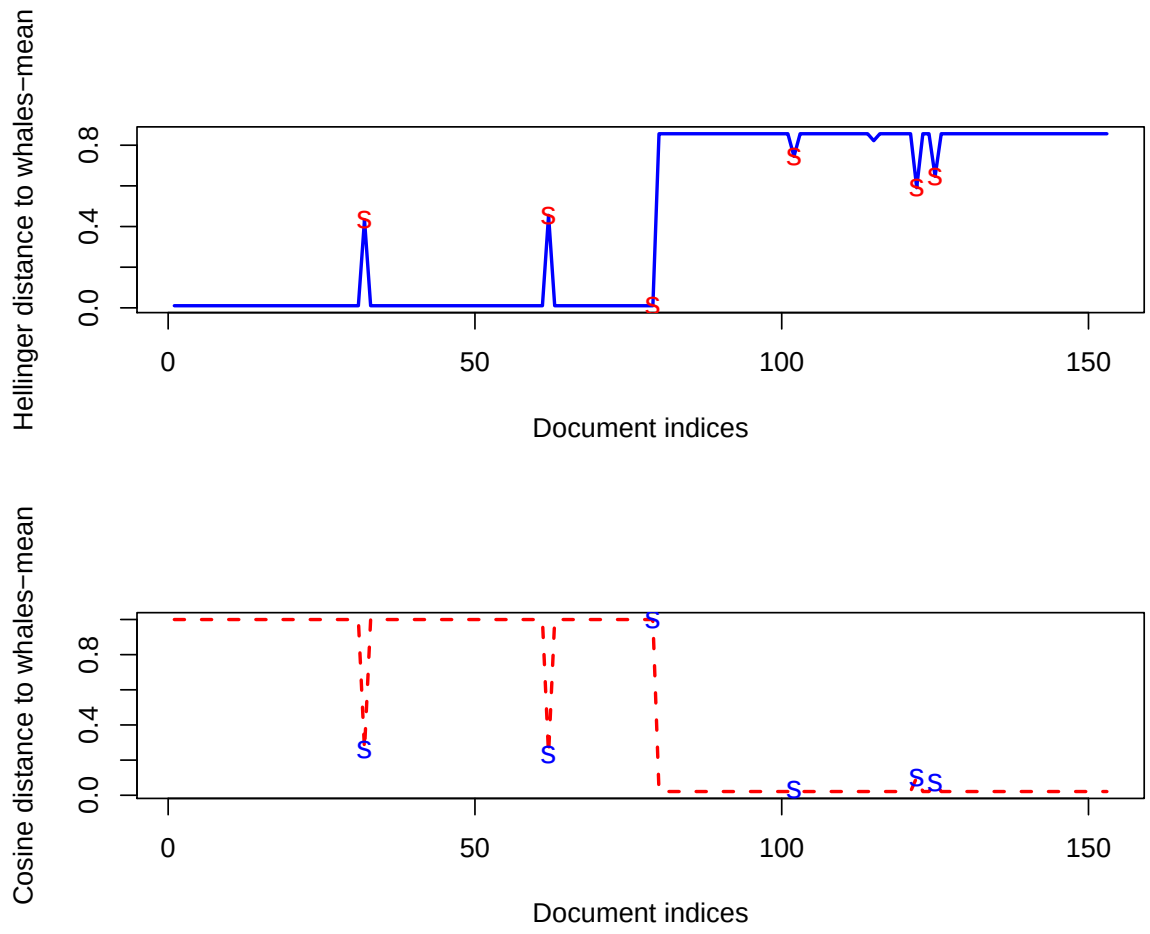

Figure 3-12. Hellinger (top) and cosine (bottom) distances of the document topic mixtures to whales' document topic mixture mean. The mark S represents support vector's Hellinger and cosine distances to whales-mean

LDA model without model retraining. A paper [29] by Yao et al. describes different methods for topic inference based on the techniques such as Gibbs sampling, variational inference, and classification. However, this thesis proposes a novel method to fit the LDA's document topic-mixtures and document word frequencies into a regression model as follows.

Firstly, to fit the regression model for document topic proportions, we reduce the vocabulary size by using the dimensionality reduction techniques described in the section 3.3.2. Secondly, we fit a multi-output regression model by fitting multiple regression models for each of the LDA topic proportions, with the document feature vectors formed

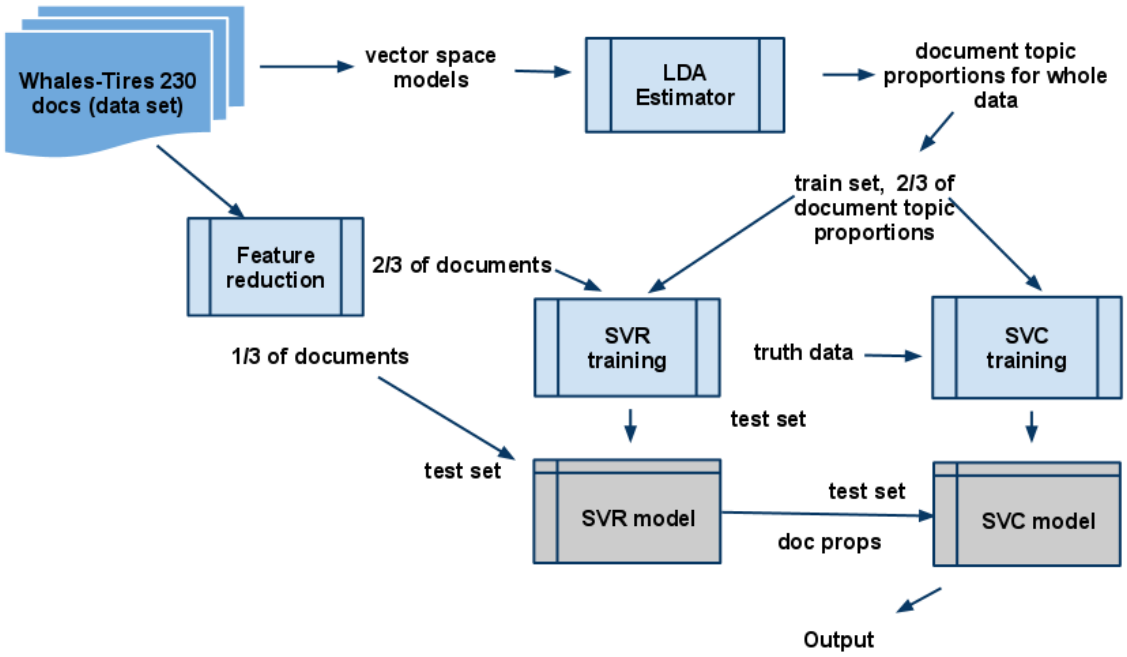

Figure 3-13. SVM as a regression and classification model with the LDA model trained on the whole whales and tires documents

by using the reduced features. In fact, for a K topic LDA model, we need K regression models to fit. Let t_i be a topic row of the K X D matrix, θ, and X be D X $\acute{V}$ matrix that represents the reduced feature vectors ($\acute{V}$) of the corpus documents. Thus, the regression model for t_i is represented as:

$$t_i = f(X, w) \tag{3–7}$$

where w represents the parameters to be learned for the regression model.

In addition, we use these learned regression models for obtaining the document topic mixture for the unseen documents. This bypasses the LDA learning process for the new documents.

Experiments: SVM for both regression and classification

In this experiment, SVM is used for both the regression (SVR) [Eq. 3–7] and classification (SVC) [section 3.3.3]. First, we see the results of an SVC model on the document topic proportions received from an SVR model using the test set, $1/3^{rd}$

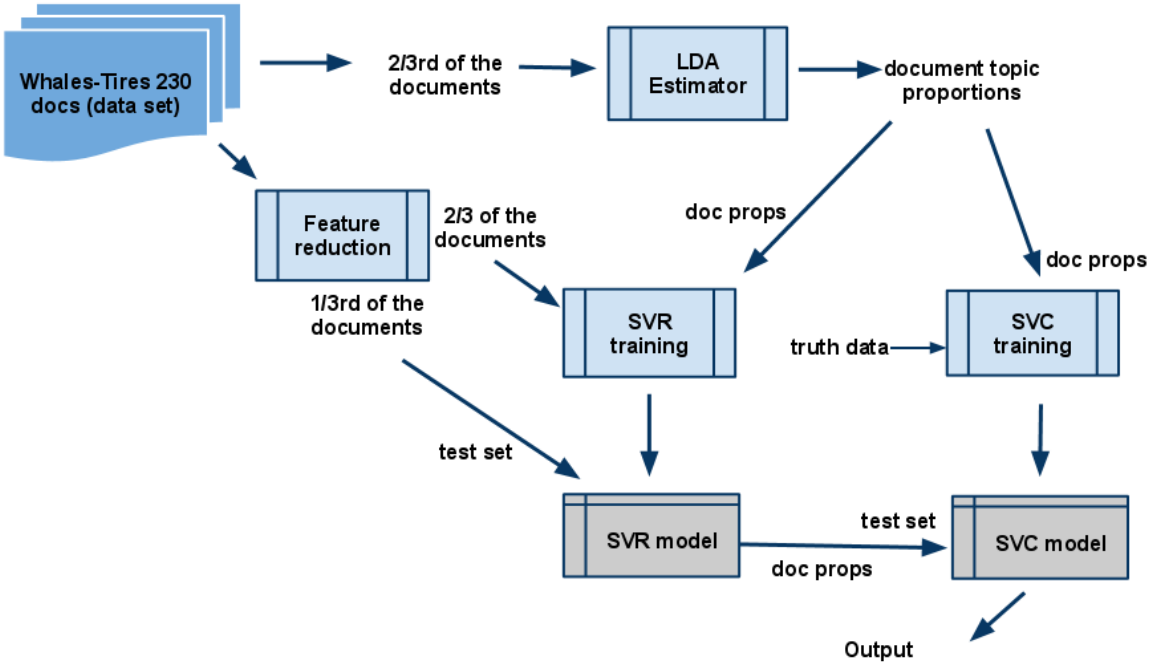

Figure 3-14. SVM as a regression and classification model with the LDA model trained on $2/3^{rd}$ of the whales and tires documents

of the whales and tires documents, [*Model 1*]. This SVR is trained on $2/3^{rd}$ of the document proportions from an LDA model learned with the whole data set i.e., 230 documents (figure 3-13). Second, we see the results of an SVC model on the document topic proportions received from an SVR model on the test set, [*Model 2*]. This SVR is trained on the document proportions from an LDA model learned with $2/3^{rd}$ of the whales and tires documents (figure 3-14).

Table 3-4 shows the classification accuracy of these two experiments compared with the LDA document proportion classifier, *Model 0* [section 3.3.3, figure 3-9]. We can see that the *Model 2* approach can be used an alternative to the classification model in which we retrain the whole data set with LDA whenever we get a new document for classification. In addition, the SVM based approach saves the storage space for models [4], because we only need to store the support vector document proportions and their parameters, which is supposedly far less than the corpus documents and their features.

Table 3-4. SVM as a regression and classification model

Model	Accuracy
SVC with the LDA's document-topic-mixture (*Model 0*)	97.40
Model 1	88.31
Model 2	84.42

3.4 The Scope of LDA in Ontology Learning

The first step in learning a realm based ontology from web pages is to find those web-pages associated with a particular domain. This thesis proposes a clustering based approach based on the topics extracted from the LDA estimation. The uncategorized web-pages are classified using an LDA based classifier [section 3.3]. The next step in ontology building is to create the ontology heterarchy. Morpheus's approach to building the ontology heterarchy is motivated by the YAGO heterarchy building process [26], which does not learn the taxonomy from text. However, it uses the WordNet [section 2.4.2] heterarchy to build the taxonomy. The classes we learned from the web pages are associated with WordNet syn-sets. For each class in the ontology we will have a class-document-model that represents a class in terms of the learned LDA model from text. The β matrix from the LDA estimation gives class term probabilities. This probabilities can be used for associating terms in a user query to the corresponding classes in an ontology. This LDA based ontology learning is in progress and it is not covered in this thesis.

3.5 Summary

This chapter gave a brief introduction to the topic model Latent Dirichlet Allocation and its possible application to modeling text documents. The highlights are using LDA for reducing the dimensionality of the corpus vocabulary and documents, and inference of document topic mixture from the learned LDA model without re-training the LDA model. In addition, this chapter compared the results of classification using the document topics proportions with varying number of topics. This chapter concludes by describing

the scope of the document modeling techniques in ontology learning from uncategorized web documents and tagging of user query terms with ontological class labels.

CHAPTER 4
QUERY RANKING

Query ranking is a challenging problem in any question answering system. Usually the user query-ranking method is based on a similarity measure defined between the user queries. This chapter describes an ontological realm-based class similarity measure and its application to query ranking. The first few sections explain *class divergence*, a quasi-metric, and a query-ranking algorithm based on it. The later sections present their implementations and results. Finally, this chapter concludes by discussing pros and cons of the class divergence measure and query-ranking algorithm.

4.1 Introduction

Morpheus solves a path follower's query, a *candidate* SSQ, by finding similar qualified SSQs that are associated with QRMs in the data store. We consider the SSQ components [section 2.3] such as query-realm, input terms and output terms, and their assigned classes to compute the similarities between SSQs.

In the Morpheus system, the classes assigned to query terms belong to a realm-ontology. We define *class divergence*, a quasi-metric, that characterizes the similarity or dissimilarity between two classes within a realm-ontology [9]. In this metric, we apply the concepts of multiple dispatch in CLOS [24] and Dylan programming [2] for generic function type matches. Once we have the similarity measures between the candidate SSQ and qualified SSQs in the QRM data store, we order the relevant SSQs and associated QRMs. The order represents a ranking for the query-results to the path follower. Subsections 4.2, and 4.3 describe this in detail.

4.2 Class Divergence

We employ a measure of compatibility, which we call *class divergence (cd)*, between two classes based on the class *topology* of an ontology. This algorithm is originally described in [9]. Let S be the source class and T be the target class. $S \prec T$ represents the reflexive transitive closure of the superclass relation. $d(P, Q)$ represents the hop distance

in the directed ontology inheritance graph from P to Q. Let C be a common ancestor class of S and T which minimizes $d(S, C) + d(T, C)$. The class divergence between S and T is defined as:

$$cd(S, T) = \begin{cases} 0 & S.Uri \equiv T.Uri \\ d(S, T)/(3h) & S \prec T \\ 1 & T \prec S \\ (d(S, root) + d(S, C) + d(T, C))/(3h) & otherwise \end{cases} \quad (4\text{–}1)$$

where h is the height of the ontology tree. The value of $cd(S, T)$ is in range of zero (representing identical classes) to one (representing incompatible classes).

In addition, if $S \prec T$ and $S \not\prec Q$ then $cd(S, T) < cd(S, Q)$. This implies that the similarity of a source class to a target ancestor class is higher than the similarity of a source class to any class that is not an ancestor [9]. This is useful in finding compatible user queries in the SSQ format. In fact, an SSQ that answers queries with an ancestor class is more relevant than an SSQ with any non-ancestral class.

Figure 4-1 illustrates the calculation of class divergence. Suppose we want to find the class divergence between *bus* and *coupe* from the ontology. *Coupe* is a subclass of *automobile* which is a subclass of *land_vehicle*. *Bus* is a subclass of *land_vehicle*, which is the least common ancestor C of *bus* and *coupe*. The longest path to the tree root from the tree leaf nodes is $h = 4$. Therefore, the normalized class divergence $cd(bus, coupe) = 6/12$. The value is calculated from $d(bus, root) = 3$, $d(bus, land\_vehicle) = 1$, and $d(coupe, land\_vehicle) = 2$.

4.3 SSQ Matching and Query Ranking

To find the relevance between the candidate SSQ and a qualified SSQ Morpheus uses *class divergence* between their assigned term classes. The qualified SSQs in the Morpheus data store contain input terms, output terms, tagged ontological classes, and a realm from the QRM. On the other hand, for the candidate query, the Morpheus NLP engine parses

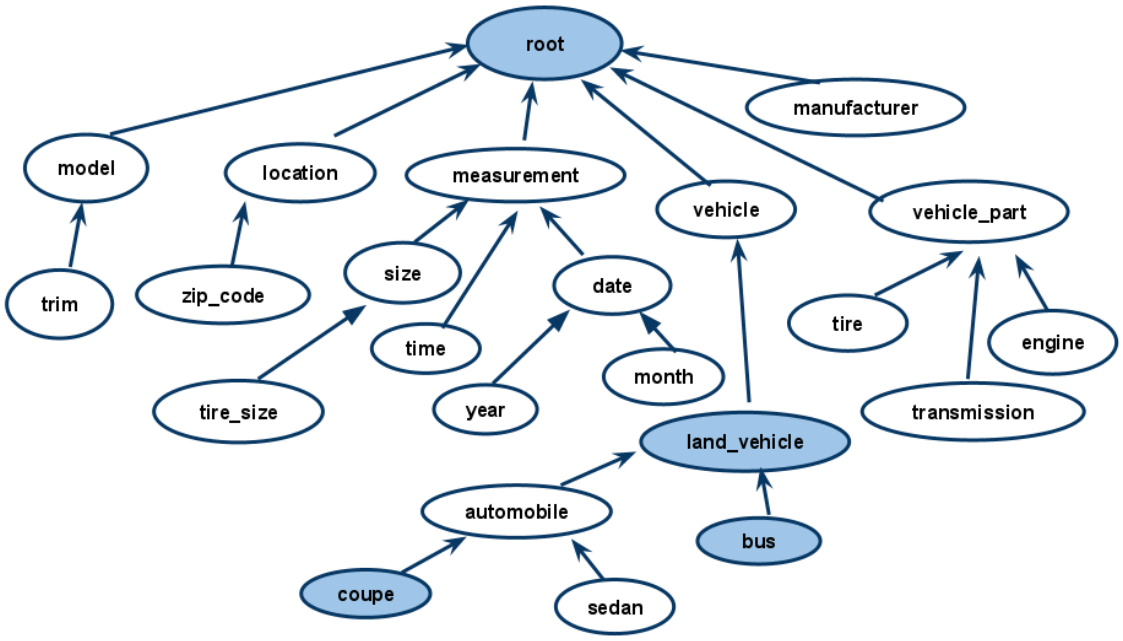

Figure 4-1. An abbreviated automotive ontology: This is a manually created ontology, rooted at the class *root*, from the Wikipedia category hierarchy and live Wikipedia pages. The arrows represent the inheritance relationships between the classes in a topology.

query-terms and assigns classes from a realm-ontology [section 3.4]. In addition, the query realm is inferred from the candidate query terms based on any probabilities calculated with $p(realm|term)$.

We refer to the sequence of these classes $\langle \Upsilon_1, ..., \Upsilon_N, \Omega_1, ..., \Omega_M, R \rangle$ as the signature of an SSQ. Υ_i represents an input class, Ω_i represents an output class, and R is for the realm of a query. If an input or output class is unspecified in the query, it is associated with the class $\perp$, which is a subclass of every other class having no subclasses or individuals.

Let $\boldsymbol{\Upsilon}$ be the set of all input classes of an SSQ and $\boldsymbol{\Omega}$ be the set of all output classes of an SSQ. Suppose we want to find the similarity of a qualified SSQ Q with signature $\langle \boldsymbol{\Upsilon}_Q, \boldsymbol{\Omega}_Q, R_Q \rangle$ to the candidate SSQ S with signature $\langle \boldsymbol{\Upsilon}_S, \boldsymbol{\Omega}_S, R_S \rangle$. Let $\langle \{\Upsilon_{Q_i}\} X \boldsymbol{\Upsilon}_S \rangle$ represents a set with elements calculated by

$$\langle \{\Upsilon_{Q_i}\} X \boldsymbol{\Upsilon}_S \rangle = \{ cd(\Upsilon_{Q_i}, b) \mid (\Upsilon_{Q_i}, b) \, \epsilon \, \{\Upsilon_{Q_i}\} X \boldsymbol{\Upsilon}_S \} \tag{4–2}$$

where $cd(\Upsilon_{Q_i}, b)$ represents class divergence between the class Υ_{Q_i} and b, and X represents the cross product of the elements of the sets $\boldsymbol{\Upsilon}_S$ and $\{\Upsilon_{Q_i}\}$.

We consider QRMs that only belong to the same realm of the candidate SSQ. To find the similarity between the qualified SSQ Q to the candidate SSQ S that belong to the same realm, we first calculate $\langle\{\Upsilon_{Q_i}\}X\boldsymbol{\Upsilon}_S\rangle$ and $\langle\{\Omega_{Q_j}\}X\boldsymbol{\Omega}_S\rangle$ for all elements of $\boldsymbol{\Upsilon}_S$ and $\boldsymbol{\Omega}_S$. The divergence of a qualified SSQ to the candidate SSQ is determined by aggregating the calculated divergence measure values as follows.

$$divergence(S, Q) = \frac{\pi_\Upsilon}{N}\sum_{i=1}^{N} min_i \langle\{\Upsilon_{Q_i}\} X \boldsymbol{\Upsilon}_S\rangle + \frac{\pi_\Omega}{M}\sum_{j=1}^{M} min_j \langle\{\Omega_{Q_j}\} X \boldsymbol{\Omega}_S\rangle \tag{4–3}$$

where N is the number of elements in the set $\boldsymbol{\Upsilon}_Q$, M is the number of elements in the set $\boldsymbol{\Omega}_Q$, and the positive weights π satisfies the condition:

$$\pi_\Upsilon + \pi_\Omega = 1 \tag{4–4}$$

Finally, we order the QRMs in the store by increasing divergence. The order provides a ranking for the results to the path follower's query. Section 2.3.1 gives a brief overview of the Morpheus query execution process using the ranked QRMs.

4.4 Results

For the illustrative purposes, we use the abbreviated automotive ontology (figure 4-1) to find the class divergence between the term-classes in this section. Suppose a path follower enters the query - "What is the tire size for a 1997 Toyota Camry V6?". The Morpheus natural language processing system parses this query into table 4-1. Table 4-2 shows a set of sample top-term-classes and their associated probabilities $p(class \mid term)$, as tagged by the NLP engine. The results shown here are based on manually constructed classes and probabilities. Our aim is to learn these class-term probabilities from the ontology and corpora built from the web documents. The n-grams with duplicated terms (marked * in table 4-2) are discarded based on the $p(class \mid term)$ values.

Table 4-1. The Morpheus NLP engine parse

Attribute	Value
WH-question Term	what
Descriptive Information	1997 Toyota Camry V6
Asking For	tires size
n-grams	1997, 1997 Toyota, 1997 Toyota Camry, Toyota, Toyota Camry, Toyota Camry V6, Camry, Camry V6, V6

Table 4-2. Top term-classes and probabilities

Term	Class	$p(Class\|Term)$
1997	year	1.00
Toyota	manufacturer	0.90
Toyota Camry*	automobile	0.60
Camry	model	0.90
Camry V6*	model	0.65
V6	engine	0.99
tire size	tire_size[1]	0.99

The final step in QRM ranking is to calculate the class divergence values for the term-classes associated with the candidate SSQ and a qualified SSQ, and aggregate them based on algorithm described in section 4.3. Table 4-3 shows divergence values calculated from the candidate SSQ to the three qualified SSQs present in the Morpheus data store. From this experiment we can infer that the aggregate class divergence for an SSQ depends mainly on the terms and tagged classes. If the assigned term-classes of both the candidate query and a qualified query are nearby in the realm-ontology the divergence values will generally be smaller, as a result, query similarity will be high. This also requires that the path finder's query must be assigned the correct term classes from the ontology, which is a challenging task.

Table 4-3. Top ranked queries based on the aggregated class divergence values

Query	Tagged Classes	Score
What is the tire size for a 1998 Toyota Sienna XLE Van?	manufacturer, model, year, tire_size[1]	0.000
Where can I buy an engine for a Toyota Camry V6?	engine, manufacturer, model, vehicle_part[1]	0.072
A 1997 Toyota camry needs what size windshield wipers?	year, manufacturer, model, measurement[1] , vehicle_part[1]	0.336

4.5 Summary

This chapter presented an ontological heterarchy based class similarity measure grounded on the concepts of type matching. It also demonstrated a prototype system to charactarize query similarity using a class divergence quasi-metric. An initial analysis shows that this is a promising method in matching and clustering the similar user queries in semi-structured format. However, thorough testing is required in some of the areas. For example, the calculation of probabilities for the user query terms and query-term class tagging are still in progress. Similarly, more research is required to include the context of the terms in a query while we search for the similar queries or SSQs.

CHAPTER 5
CONCLUSIONS

This thesis started by describing the challenges of building a question answering system and a realm-based approach in particular. Subsequently, it presented the idea of using an ontology and associated corpora to represent the web in a more hierarchical form helping the Morpheus QA system to match similar queries in the SSQ format. The core idea is to represent user query terms by a set of classes from a realm-based ontology and find similarities in the space of ontological classes. In addition, we calculate class similarity as a type match based on the class taxonomy of the realm-ontology. We can extend this idea of finding class similarities to any systems that stores information in a class heterarchy. Assigning classes from a realm-ontology also helps clustering natural language queries that belong to a local realm context.

In addition, we considered the challenging task of building an ontology and associated corpora from the web which often stores information in unstructured and uncategorized form. In order to tackle this problem, the thesis built frameworks for categorizing and extracting hidden topics from web pages. We discussed the well known topic model Latent Dirichlet Allocation to detect topics from text. We observed that it is useful for distinguishing those domains that are far apart, e.g., whales and tires. However, it fails to distinguish documents when they are from similar domains, e.g., whale types. In addition, we saw that LDA is useful in reducing dimensionality of documents into a topic space and building classifiers on the topic space. This LDA-based dimensionality reduction can be applied to any system where one can assume the existence of a topic space on top of the feature space.

Moreover, we saw a promising method that can avoid LDA retraining for the corpus documents, by fitting a multi-output regression model based on Support Vector Regression using the document-term-frequency feature vectors and the LDA topic proportions. This is a useful in systems where we need to apply LDA on streaming documents. Even though

LDA is seen to be promising approach to classification and topic extraction, more research is required to find the applicability of LDA topics in learning an ontology taxonomy from a plain text corpus.

REFERENCES

[1] Auer, Sören, Bizer, Christian, Kobilarov, Georgi, Lehmann, Jens, and Ives, Zachary. "DBpedia: A Nucleus for a Web of Open Data." *In 6th Intl Semantic Web Conference, Busan, Korea*. Springer, 2007, 11–15.

[2] Barrett, Kim, Cassels, Bob, Haahr, Paul, Moon, David A., Playford, Keith, and Withington, P. Tucker. "A monotonic superclass linearization for Dylan." *SIGPLAN Not.* 31 (1996).10: 69–82.

[3] Bird, Steven, Klein, Ewan, and Loper, Edward. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. Beijing: O'Reilly, 2009.

URL http://www.nltk.org/book

[4] Bishop, Christopher M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007, 1st ed. 2006. corr. 2nd printing ed.

URL http://www.worldcat.org/isbn/0387310738

[5] Bizer, Christian, Lehmann, Jens, Kobilarov, Georgi, Auer, Sören, Becker, Christian, Cyganiak, Richard, and Hellmann, Sebastian. "DBpedia - A crystallization point for the Web of Data." *Web Semantics: Science, Services and Agents on the World Wide Web* 7 (2009).3: 154–165.

URL http://linkinghub.elsevier.com/retrieve/pii/S1570826809000225

[6] Blei, David M., Ng, Andrew Y., and Jordan, Michael I. "Latent dirichlet allocation." *Journal of Machine Learning Research* 3 (2003): 993–1022.

[7] Chang, Chih-Chung and Lin, Chih-Jen. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[8] Dohzen, Tiffany, Pamuk, Mujde, Seong, Seok W., Hammer, Joachim, and Stonebraker, Michael. "Data integration through transform reuse in the Morpheus project." *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2006, 736–738.

URL http://dx.doi.org/10.1145/1142473.1142571

[9] Grant, Christan, P. George, Clint, Gumbs, Joir-dan, Wilson, Joseph N., and Dobbins, Peter J. "Morpheus: A Deep Web Question Answering System." *International Conference on Information Integration and Web-based Applications and Services*. Paris, France, 2010.

[10] Green, B, Wolf, A, Chomsky, C, and Laughery, K. "BASEBALL: an automatic question answerer." (1986): 545–549.

[11] Griffiths, T. L. and Steyvers, M. "Finding scientific topics." *Proceedings of the National Academy of Sciences* 101 (2004): 5228–5235.

[12] Gruber, T. "A translation approach to portable ontology specifications." *Knowledge Acquisition* 5 (1993).2: 199–220.

URL http://linkinghub.elsevier.com/retrieve/doi/10.1006/knac.1993.1008

[13] Hahn, Rasmus, Bizer, Christian, Sahnwaldt, Christopher, Herta, Christian, Robinson, Scott, Bürgle, Michaela, Düwiger, Holger, and Scheel, Ulrich. "Faceted Wikipedia Search." *BIS*. 2010, 1–11.

[14] Hofmann, Thomas. "Probabilistic latent semantic indexing." *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 1999, 50–57.

[15] Horowitz, Damon and Kamvar, Sepandar D. "The anatomy of a large-scale social search engine." *WWW '10: Proceedings of the 19th international conference on World wide web*. New York, NY, USA: ACM, 2010, 431–440.

[16] Lin, Jimmy and Katz, Boris. "Question answering from the web using knowledge annotation and knowledge mining techniques." *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*. New York, NY, USA: ACM, 2003, 116–123.

[17] Liu, Yandong and Agichtein, Eugene. "On the evolution of the yahoo! answers QA community." *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2008, 737–738.

[18] Nguyen, Hoa, Nguyen, Thanh, and Freire, Juliana. "Learning to extract form labels." *Proc. VLDB Endow.* 1 (2008).1: 684–694.

[19] Noy, N. F. and Mcguinness, D. "Ontology Development 101: A Guide to creating your first Ontology." *Stanford KSL Technical Report KSL-01-05* (2000).

URL http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html

[20] Pearl, Judea. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[21] Pessiot, Jean-Franois, Kim, Young-Min, Amini, Massih R., and Gallinari, Patrick. "Improving document clustering in a learned concept space." *Information Processing and Management* 46 (2010).2: 180 – 192.

[22] Schlaefer, Nico, Gieselmann, Petra, and Schaaf, Thomas. "A pattern learning approach to question answering within the Ephyra framework." *Proceedings of the Ninth International Conference on TEXT, SPEECH and DIALOGUE*. 2006.

[23] Srivastava, Ashok and Sahami, Mehran. *Text Mining: Classification, Clustering, and Applications*. Chapman & Hall/CRC, 2009.

[24] Steele, Jr. Guy L. *Common LISP: the language (2nd ed.)*. Newton, MA, USA: Digital Press, 1990.

[25] Steyvers, Mark and Griffiths, Tom. *Probabilistic Topic Models*, chap. Latent Semantic Analysis: A Road to Meaning. Laurence Erlbaum. Lawrence Erlbaum Associates, 2007.

URL http://www.worldcat.org/isbn/1410615340

[26] Suchanek, Fabian M. *Automated Construction and Growth of a Large Ontology*. Ph.D. thesis, Saarland University, 2009.

URL http://www.mpi-inf.mpg.de/~suchanek/publications/submitted.pdf

[27] Wei, Xing and Croft, W. Bruce. "LDA-based document models for ad-hoc retrieval." *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2006, 178–185.

[28] Woods, W. "Progress in natural language understanding - an application to lunar geology." *American Federation of Information Processing Societies (AFIPS) Conference Proceedings, 42*. 1973, 441–450.

[29] Yao, Limin, Mimno, David, and McCallum, Andrew. "Efficient methods for topic model inference on streaming document collections." *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2009, 937–946.

[30] Zhang, Zhiwei, Phan, Xuan-Hieu, and Horiguchi, Susumu. "An Efficient Feature Selection Using Hidden Topic in Text Categorization." *22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008)* (2008): 1223–1228.

URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4483086

BIOGRAPHICAL SKETCH

Clint Pazhayidam George received his *Bachelor of Technology* in Computer Science from the Department of Computer Science and Engineering, *College of Engineering Trivandrum*, University of Kerala, India in 2004. After graduation he worked for four years as a software professional. He graduated with his *Masters* in Computer Science from the Department of Computer & Information Science and Engineering, *University of Florida*, Gainesville, Florida in 2010. His research interests include machine learning, applied statistics, text mining, and pattern recognition.