Automatic Document Topic Identification Using Hierarchical Ontology Extracted from Human Background Knowledge

by

Mostafa Hassan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The rapid growth in the number of documents available to various end users from around the world has led to a greatly increased need for machine understanding of their topics, as well as for automatic grouping of related documents. This constitutes one of the main current challenges in text mining.

We introduce in this thesis a novel approach for identifying document topics. In this approach, we try to utilize human background knowledge to help us to automatically find the best matching topic for input documents. There are several applications for this task. For example, it can be used to improve the relevancy of search engine results by categorizing the search results according to their general topic. It can also give users the ability to choose the domain which is most relevant to their needs. It can also be used for an application like a news publisher, where we want to automatically assign each news article to one of the predefined news main topics. In order to achieve this, we need to extract background knowledge in a form appropriate to this task. The thesis contributions can be summarized into two main modules.

In the first module, we introduce a new approach to extract background knowledge from a human knowledge source, in the form of a knowledge repository, and store it in a well-structured and organized form, namely an ontology. We define the methodology of identifying ontological concepts, as well as defining the relations between these concepts. We use the ontology to infer the semantic similarity between documents, as well as to identify their topics. We apply our proposed approach using perhaps the best-known of the knowledge repositories, namely Wikipedia.

The second module of this dissertation defines the framework for automatic document topic identification (ADTI). We present a new approach that utilizes the knowledge stored in the created ontology to automatically find the best matching topics for input documents, without the need for a training process such as in document classification. We compare ADTI to other text mining tasks by conducting several experiments to compare the performance of ADTI and its competitors, namely document clustering and document classification. Results show that our document topic identification approach outperforms several document clustering techniques. They show also that while ADTI does not require training, it nevertheless shows competitive performance with one of the state-of-the-art methods for document classification.

Acknowledgements

First and foremost, all thanks and praise is due to The Great Almighty Allah for granting me the strength, guidance, blessing, and knowledge, that enabled me to complete this work.

This thesis would not be possible without the support of many individuals, to whom I would like to express my gratitude.

I will always be indebted to my supervisors, Prof. Fakhri Karray and Prof. Mohamed Kamel, for their support, encouragement, guidance, and most importantly trust. Prof. Karray's trust and support was instrumental in giving me confidence to achieve many accomplishments. I would like to thank Prof. Karray for his encouragement and guidance throughout my research. Prof. Kamel's input and guidance was invaluable to the quality and contribution of the work presented in this thesis, as well as in other publications. I would like to thank Prof. Kamel for his advice, valuable insights and feedback throughout my research. Without them this research would not have been possible.

I would like also to thank many faculty members of the University of Waterloo, most notably my committee members, Prof. Chrysanne DiMarco, Prof. Krzysztof Czarnecki and Prof. Zhou Wang as well as my external examiner Prof. Reda Alhajj for taking the time to assess my research work, as well as for their pertinent advice and feedback.

I wish to thank many of my colleagues at the Centre of Pattern Analysis and Machine Intelligence (CPAMI) and University of Waterloo, especially Ahmed Elgohary, Haitham Amar, Nabil Drawil, Abbas Ahmadi, and Mehrdad J. Gangeh for valuable discussions and insights.

I also would like to thank Ahmed K. Farahat for being such a great friend and for his constant support and help during my studies. I would like also to thank Ahmed Elmogy, Abdulrahman Seleim, Mohamed Hamouda, Moataz El Ayadi, Ahmed Othman, Mohamed AbdelRazik, Ahmed Gawish, and Shady Shehata for being such great friends. I would like also to extend my appreciation to my friends in Egypt and Canada for their support and encouragement. Not forgetting to thank my previous supervisor, Prof. Amir Atiya, who led me into the field of data mining and gave me kind guidance in both of my Master and Bachelor degrees.

My family is the most precious treasure in my life. They motivated and encouraged me unfailingly. Many thanks to my wife for her optimism, and her resolve. This thesis is as much a product of her unconditional support and patience, as it is of my efforts. Words fail me to express my appreciation to her. I would like to thank my son, Abdulrahman, for brightening my life. I wish also to thank my brother, my sisters, and my whole family for

their continued support, patience, and prayers. Last and by far not least, a huge thanks to my parents, who taught me the value of a good education. Thanks to them for their unlimited support and encouragement throughout my life, and for everything that they offered to me without my even needing to ask. Therefore, my deepest gratitude goes to them, and I dedicate this thesis to my father, mother, and all my family.

To ***My Gracious Parents, My Wife and My Son.***

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The increasing demand for organizing and gathering knowledge from large numbers of documents has led to a corresponding demand for better text processing tools. Text processing is the set of techniques and tools that are used to extract information from text. There are many different domains for text processing. Grobelnik [2] has classified the text processing module into four different active domains as shown in Figure 1.0.1. Information retrieval (IR) is the process of gathering information, usually in the form of a document, that matches a required information need, in the form of a query, from a collection of information resources. Search engines are a good example of an application of IR. This process includes all the sub-tasks, starting from modeling documents, through retrieval of matching documents, to measuring performance of this process. Natural language processing (NLP) is the field of computer science that is concerned with handling human natural language. Natural language understanding and computational linguistics are the most common sub-fields of NLP.

Knowledge representation (KR) is the area of artificial intelligence that is concerned with extracting and representing knowledge to be used in different text processing approaches. Text mining is the process of gathering well-structured information from unstructured text documents. In other words, it applies different data mining techniques to text documents to derive patterns and trends. Unlike data mining, which accesses data in a raw and easily-manipulable numeric format, text mining often deals with amorphous and hard-to-manipulate text documents (e.g., news, email messages, spoken documents, etc.).

The text mining process can be divided into four main steps. The first step is document pre-processing, in which the text document is parsed and tokenized into terms, and the stop words are removed from the output terms. Next is the document modeling step, in which documents are converted from the textual representation to a numerical format so

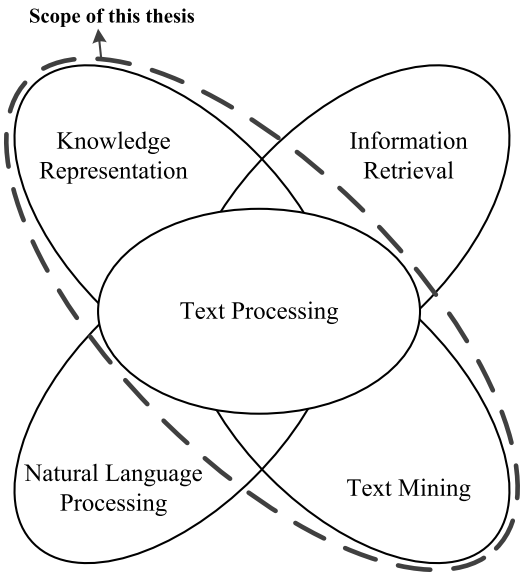

Figure 1.0.1: Text Processing active domains

that they can be handled in the next step. Third, the derivation of patterns and trends step, where a data mining technique (such as classification and clustering) is applied to this numerical representation of the documents to accomplish a specific data-mining task (spam filter, topic prediction, summarization). The last step is to measure the performance of the previous steps, using several available metrics.

Usually, any text processing task consists of a combination of two or more tasks from different domains. In our proposed work, we introduce the use of some knowledge representation techniques to improve the performance of some text mining tasks. The scope of the thesis is grouped in the dashed ellipse area in Figure 1.0.1.

Recently, many approaches in the literature employ the use of background knowledge to improve performance of various text mining tasks. In our proposed work here, we introduce a new approach to extract this background knowledge from any knowledge repository and store it in a well-structured, organized form, which is easy to apply to text mining tasks. Of course, this knowledge repository must have certain features to be usable for our approach. We refer to our created form of background knowledge as an ontology. An ontology in text mining is defined as any knowledge representation consisting of a set of concepts whose definitions and interrelations are explicitly described. We apply our proposed approach to one of the best-known knowledge repositories, namely Wikipedia. We use our created ontology to infer the semantic relatedness between terms, as well as to identify the topics

of the documents.

Document topic identification - or, in short, topic identification - is usually used to refer to the task of finding relevant topics in a set of input documents[3, 4]. These topics are usually extracted from an external background knowledge base. This definition of topic identification is similar to the definition of the document tagging task, where more than one topic is assigned to each input document. Sometimes, we have a set of topics that has been marked as being "of interest", and we want to assign each input document to one of these topics. For instance, if we are interested in the following topics: economics, politics, and sports, and we have a document declaring "Barack Obama is the new President of the United States", we want to identify the most relevant topic as "politics". This task is usually needed for an application like a news publisher, where we want to automatically assign each news article to one of the predefined news topics. We refer to this task as Automatic Document Topic Identification (ADTI). In this work, we present a new approach that utilizes the knowledge stored in our ontology to automatically identify document topics. The main goal of this application is to identify an input document's topic from a set of input topics with the help of the background knowledge - without the need of training.

1.1 Motivations

The main motivation of building the hierarchical ontology is that we are trying to imitate, in some sense, the way that humans learn about a new subject. For example, when we read an article that talks about "document modeling", we know that this subject falls under the topic of "text mining" or "information retrieval", and we know also that these topics are under a bigger category called "computer science", which is in turn under the "science" category. This hierarchical representation of human knowledge led us to use hierarchical knowledge representation.

Also, when we read more articles on these topics, we learn that a term like "Vector Space Model" frequently co-occurs with terms like "bag of words", "document modeling", "Salton", and so on. Although these terms are not synonymous to one another, they are *semantically related* and the co-occurrence of these terms in a document indicates that the topic of it is in the field of text mining or information retrieval. This is what we want to store in our ontology. Here we give a motivating example which points out the importance of inferring the semantic relationship between terms:

Example 1: Consider the following sentences:

- "Securing a more balanced global economy",

- "The *Unemployment* rates are going down all over the world",
- "The *government* wants to see more universities accorded such autonomy",
- "Most certainly, the direction of world *politics* has been changed", and
- "Barack Obama is the new *president* of the U.S."

Let us suppose that these represent five different short documents, and we want to measure the similarity between these documents. In conventional document modeling techniques such as the VSM, the similarity between these documents is nonexistent, because there are no terms in common amongst them.

Although this is objectively true, human beings still could identify some similarity between these sentences. For example, we can easily find some correlation between the first two sentences. This is because the terms "economy" and "unemployment" have some embedded relation to one another. This relation is referred to as "semantic relatedness". Hence, we could find similarity between documents even if they do not have any terms in common. We refer back to this example in section 3.4.2.

1.2 Goals

The main goal of the proposed approach is to build an expert document-manipulating system. It should be able to identify the document topics by assigning each document to a topic from a given set of input topics. We refer to this as Automatic Document Topic Identification (ADTI), which is described in detail in chapter 4. To fulfill these requirements, we break this task down into the following two main tasks of our proposed approach:

1. Building a knowledge structure which covers a large portion of human knowledge. This structure should be organized in such a way that it can be easily used for different tasks in text mining.

2. Using the extracted knowledge structure to build topics representation, hence identifying the best matching topic for input documents.

1.3 Thesis Outline

After a brief introduction and discussion of the motivations and goals of the research, the organization of the thesis will be as follows: In chapter 2, we start by discussing some

of the background material about text mining tasks and modules that are related to our proposed work. This includes an overview of the basic document representation model, and a review of some of the text mining tasks, such as document clustering and document topic indexing. Then we describe what is meant by an ontology, and how it is used for different text mining tasks. After that, we emphasize studying those methods in the literature which use background knowledge for text mining tasks.

In chapter 3, we introduce the process of extracting background knowledge from any knowledge repository. We also show the requirements that must be met by any knowledge repository, to be able to extract its knowledge in the form of an ontology. Then we demonstrate how to apply the proposed knowledge extraction approach to Wikipedia, to create the Wikipedia Hierarchical Ontology, WHO. This includes the discussion of the reason for choosing Wikipedia, and how we evaluated the performance of WHO.

Chapter 4 presents a novel approach for topic indexing, the Automatic Document Topic Identification (ADTI) approach. We start by giving a brief introduction to this approach, then describe it in detail, discussing the different modules and the aspects that affect its performance. After that, we conduct a comparison between this approach and the similar text mining tasks.

In chapter 5, we provide simulations and results of applying ADTI using the proposed WHO, and compare its performance to the performance of both conventional document clustering and document classification techniques. Finally, chapter 6 concludes the work and proposes a plan for further research.

Chapter 2

Background and Literature Review

2.1 Overview

As noted above, we are proposing in this research a method for extracting the background knowledge from one of the knowledge repositories, and storing it in a form that can be used in different text mining tasks, namely an ontology. In order to cover the related background issues and the state of the art, we start by giving a review of different text mining tasks that are related to our proposed research work. This includes document classification, document clustering, and document topic indexing. Then we cover the definition of the ontology, and review the different approaches that employ ontologies in text mining, in the literature. Since we have used Wikipedia as a knowledge source to create the ontology, we have also surveyed a portion of the research done in the literature which utilizes Wikipedia as a source of background knowledge to accomplish different text mining tasks.

2.2 Text Mining

Text mining is the process by which we deal with a collection of documents to extract some useful information. In contrast with data mining, text mining deals with the data in an unstructured, textual format, such as is found in document collections. The term "text mining" was first proposed in 1995 by Feldman and Dagan [5]. As we pointed out in chapter 1, text mining consists of four different modules. The first step is document preprocessing. The second step of the text mining process is the conversion of the input data from its raw format to a structured, easy-to-manipulate format. This module is called Document

Modeling, which we discuss in detail in section 2.2.1. Another module in text mining is Deriving Patterns and Trends, in which there are some predefined tasks that extract different types of information from text documents. Examples of these tasks are: Document Clustering, Document Classification, Document Summarization, Document Tagging, and Document Topic Identification. In subsections (2.2.2.2 and 2.2.2.3), we discuss some of these tasks in detail. The last module in text mining is Measuring the Performance, which we review in section 2.2.3.

2.2.1 Document Representation Models

In data mining, the first step to deal with a given problem is to convert the data from its raw format to a compact representation which is easier to manipulate. This step is called feature extraction or data modeling. In text mining applications, the documents need to be represented in such a way that this representation reflects the meaning of the document. So, the selection of a representation for text depends on what one considers as the meaningful text units. These meaningful text units or "text features" are called terms, thus the document is represented as a set of terms.

There are many different document representation methods which generate different types of terms for documents. The choice of the term set has a huge effect on the quality of the text mining process overall. The most common basic approach for document representation is the Bag-Of-Words model (BOW), where the term is identified as a single word. This assumes that there is no significance for grammar, or even word order, in the document.

The Boolean model is an example of a simple retrieval model. In this model, each term in the document is associated with naught, which means that the term does not appear in the document, or with unity, which means that the term appears in the document at least once. The main disadvantage of this model is that if there are no common terms between two documents, there will be no similarity between them. Sometimes this is not true and there should be a partial similarity between them.

There are other document representation models based on the BOW's basic idea. One of these models that is very popular in the literature of information retrieval is the Vector Space Model (VSM). This was first proposed by Salton et al. [6] in 1975. In the following subsection, we discuss this model in detail.

2.2.1.1 Vector Space Model (VSM)

It was realized that the use of the Boolean model for document representation was insufficient, and that there was a need for a model that would allow for partial matching between documents. There was also need of a method to measure the degree of the similarity between these documents. The vector space model was proposed to achieve these requirements, after being introduced by Salton et al. [6] in the 1970s. Currently, this model is widely used in information retrieval and text mining tasks.

In the vector space model, documents are modeled as vectors in a vector space. Each dimension corresponds to a separate term, but instead of assigning each term a Boolean value to represent whether it exists or not, each term is associated with a certain weight. This weight represents how this term contributes to the meaning of the document. This implies that all the algebraic rules and operations for vectors can be applied to the documents. Therefore, if we have a set of p documents $\{d_i : \ i = 1, \ldots, m\}$, each document d_i is represented as:

$$\overrightarrow{d_i} = (w_{i1}, w_{i2}, \ldots, w_{in})$$

where n is the total number of terms. The Vector Space Model was mainly proposed to evaluate the degree of similarity between two documents from the closeness between the vectors representing these documents. Although dot-product can be used to measure the correlation between vectors, the cosine of the angle between the vectors is preferable as it is a normalized measure and is not affected by document length. So, the similarity, S_{ij}, between a document d_i and a document d_j can be defined as follows:

$$\begin{aligned} S_{ij} &= \frac{\overrightarrow{d_i} \bullet \overrightarrow{d_j}}{\left|\overrightarrow{d_i}\right| \times \left|\overrightarrow{d_j}\right|} \\ &= \frac{\sum_{k=1}^{n} w_{ik} \times w_{jk}}{\sqrt{\sum_{k=1}^{n} w_{ik}^2} + \sqrt{\sum_{k=1}^{n} w_{jk}^2}} \end{aligned}$$

where $\overrightarrow{d_i} \bullet \overrightarrow{d_j}$ is the dot product between the document vectors and $\left|\overrightarrow{d_i}\right|, \left|\overrightarrow{d_j}\right|$ are the norms of the document vectors. This model implicitly considers terms as orthogonal vectors

in an n-dimensional Cartesian space, $\mathbb{R}^n$. Wong et al. [7] stated that without this assumption, it is not possible to characterize the vector space completely. The above similarity measure can be expressed in a matrix form as follows:

$$S = L^{-1} A A^T L^{-1}$$

where A, which is known as the Document-Term matrix, is defined as

$$A = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix} \tag{2.1}$$

where each row in A represents a document, each column represents a term, AA^T represents the dot product between all documents' vectors, and L is defined as:

$$L = \sqrt{diag\left(\sum \left(A^T\right)^2\right)}$$

The Vector Space Model can be used with different term representation. Single-word representation is the most commonly-used approach for term representation, where each term is represented as a single word. While this method is direct and quite simple, it is very powerful in document representation. There are various forms of term representation proposed in the literature. The most common approaches are n-grams of words, n-grams of characters, and phrases, each of which is used in different applications [8]. In the phrases approach, documents are represented in terms of their important phrases. Different approaches have been proposed to extract these phrases from documents. One of these approaches is the Document Index Graph (DIG), which is based on finding the matching phrases between documents[9, 10, 11].

There are various methods to evaluate the weight that should be associated with each term in the document. This weight should reflect the importance of that term to the meaning of the document. A simple method to calculate this weight is the term count, which is defined as:

$$\mathrm{tc}_{ij} = n_{ij}$$

where n_{ij} is the number of occurrences of the term t_i in document d_j. The main problem with this measure is that the weight of the term is affected by the document length. In other words, if term t_1 occurs n_{11} times in document d_1 and n_{12} in document d_2, and $n_{11} > n_{12}$, this does not necessarily mean that t_1 is more important to d_1 than to d_2; it may be that the size of d_1 is much greater than the size of d_2. The term frequency weight is the normalized version of the term cont weight, where tf is defined as:

$$\mathrm{tf}_{ij} = \frac{\mathrm{tc}_{ij}}{\text{size of } \mathrm{d}_i} = \frac{n_{ij}}{\sum_{k=1}^{n_j} n_{kj}}$$

where n_j is the total number of terms in document d_j. This measure is considered to be a local measure, as it represents how important (discriminative) the term t_i is to the document d_j, rather than the other terms in this document. There is another weight which evaluates how discriminative the term t_i is to the document d_j rather than to other documents, which is the inverse document frequency. It reflects the global importance of the term. It is defined as:

$$\mathrm{idf}_i = \frac{m}{m_i}$$

where m is the total number of documents, and m_i is the number of documents that contains the term t_i. If a term occurs in numerous documents, it implies that it is not that important to all documents, similar to stop words, hence it will have a small value of idf. The combination of the previous two measures is called tf-idf. It is defined as:

$$\text{tf-idf} = \mathrm{tf} \times \log\left(\mathrm{idf}\right).$$

The logarithm is used here to decrease the effect of the inverse document frequency factor. This measure is the most commonly used weighting function. It combines the local importance of the term to the document with the global discrimination weight of this term to the document.

Shortcomings of VSM As noted above, VSM assumes that the terms' vectors are orthogonal. Since the terms are really correlated, this assumption is not true, which affects the performance of the model. For example, terms like "Economy" and "Unemployment" are semantically correlated. So, when a term like "Economy" occurs in one document and another term like "Unemployment" occurs in another document, this implies that these two documents are semantically correlated. In VSM, "Economy" and "Unemployment" terms are represented as orthogonal vectors. Thus, these two terms are considered uncorrelated.

2.2.2 Deriving Patterns and Trends

In this section, we review some of the well-known approaches which are related to our proposed work and are used in text mining to derive patterns and trends.

2.2.2.1 Document Classification

Statistical classification is the process of finding the best matching category (class) for each new observation on the basis of a set of training observations whose category membership is given. Observations are represented in a form of a set of properties which are called features. The algorithm that implements the classification process is called a classifier. Given a set of categorized observations, the goal of a classifier is to find the pattern that links the observations that belong to the same category together and differentiate between the observations that belong to different categories, then to apply the learned pattern to new observations. Hence, classification can be considered a supervised learning approach as it needs a training set, a set of correctly identified categories, to learn the pattern from as opposed to the unsupervised learning approaches - such as clustering, which groups data points into clusters based on their similarity without any given categorization. We will discuss clustering in detail in the next section.

Accordingly, document classification is the process of assigning each document a category (class) based on some labeled documents. Usually, documents are represented as vectors in the term space, as we discussed in section 2.2.1. Documents can be classified according to different aspects, such as their topics, size, author, and so on. They can also be classified to a more generic form of topic, such as spam and non-spam for emails. Unless it is explicitly mentioned, document classification usually refers to document topic classification.

There are many different algorithms and techniques in the literature for classification generally, and for document classification specifically. These algorithms include k-nearest neighbour classifier, nearest centroid classifier, naïve Bayes classifier, Bayesian networks, neural networks, and support vector machines. One of the most challenging issues in document classification is document representation. Documents usually contain a large number of features, typically thousands of terms. Also, documents' feature vectors are very sparse. Both of these issues affect the performance of the classifier dramatically. The most well-known approach that can handle these document classification issues is the Support Vector Machines classifier technique, SVM. In the next subsection, we will discuss the support vector machines approach in more detail. Afterwards, we will discuss the nearest centroid classifier, as it has some similarity with our proposed approach in chapter 4.

Support Vector Machines Classifier The support vector machines classifier is one of the most commonly used classification algorithms in text mining. It was originally proposed by Cortes and Vapnik in 1995 [12]. Its main strength for text mining is that it can handle the sparsity of the document representation. It is also proven to be the state-of-the-art method for document classification [13]. The basic SVM is a binary linear classifier, which means that it can only handle data sets with two classes, labeled 1 and -1.

Its basic idea is to use the training examples to create a linear decision boundary which separates the training samples of the two classes with a gap, which should be as wide as possible. This boundary is represented in a form of a separating hyperplane which aims to minimize the total classification error. This hyperplane is defined by a set of support vectors which are a subset of the training set, that occur on the boundaries between the two classes. Then in the class prediction phase, it assigns each new sample to one of the two classes based on its position relative to the decision boundaries.

Usually, data sets are not linearly separable. Thus, SVM can handle the non-linear separable data by mapping the data set into a high dimensional feature space where the data set classes will be linearly separable. This mapping operation is called the kernel trick. SVM can also be extended to handle multiple classes. There are two different approaches for this extension. The first approach is to reduce the single multi-class problem into multiple binary classification problems. This approach includes building binary classifiers which distinguish between one class against the rest or between every pair of classes [14, 15], directed acyclic graph SVM[16], and error-correcting output codes[17]. The other approach is to cast the multi-class SVM method into a single optimization problem, which was proposed by Crammer and Singer[18].

Nearest Centroid Classifier Nearest centroid classifier, also called nearest prototype classifier, is a classification algorithm which assigns each new observation to one of the classes based on the similarity between the new observation and the classes' prototypes. The nearest prototype to the observation corresponds to the class that the observation should be assigned to. Prototypes are usually evaluated by calculating the mean value of each class of observations, hence the name centroid. Although fairly simple, Han and Karypis have shown that this classification outperforms other algorithms such as naïve Bayes, k-nearest neighbour, and C4.5 classifiers[19]. When this classifier is applied to documents, it is also referred to as the Rocchio classifier, since it is very similar to the Rocchio algorithm for relevance feedback. The nearest centroid classifier algorithm in text mining is very straightforward; it starts with a training phase, where classes' centroids are measured by calculating the vector summation of all documents' vectors that have the same class label. Then in the class prediction phase, the cosine similarity between each

class centroid and the test document vector is measured, and the document is assigned to the nearest centroid vector class.

2.2.2.2 Document Clustering

Clustering is the process of assigning each input pattern to a group (cluster), such that each group contains similar patterns. The similarity measure depends on the nature of these patterns. Accordingly, text document clustering is the process of grouping text documents into groups of similar documents. The similarity in document clustering is measured by how the documents are semantically related to each other. It depends on the document representation model. The similarity between documents can be represented as a distance measure.

In cluster analysis, there are different distance measures defined to calculate the similarity, such as Euclidean distance, Manhattan distance, Mahalanobis distance, cosine distance, and Hamming distance. Cosine distance is often used when clustering high dimensional data, such as those derived from documents - hence the most commonly used distance measure in document clustering is cosine distance.

Data clustering algorithms can be classified into different categories based on the techniques used for clustering. Some of these algorithms are based on statistical analysis such as Hidden Markov Models (HMM) or Expectation Maximization (EM) [20], while other algorithms are based on Neural Networks such as Self Organizing Maps [21]. Also, they can be categorized based on the technique for structuring clusters into hierarchical clustering and partitional clustering.

Hierarchical clustering Hierarchical clustering algorithms are the category of clustering algorithms in which a hierarchy of clusters is created in a tree-like structure called a Dendrogram. The root of the tree represents one cluster which contains all the data points (documents). The number of the leaves of the tree is equal to the total number of data points (documents) and each leaf represents a cluster which corresponds to a single data point (document).

A hierarchical clustering algorithm is either bottom-up (*agglomerative*) or top-down (*divisive*). In an *agglomerative* algorithm, it starts with each data point in one cluster. Then, it selects the two clusters with the strongest link and combines them. The strongest link here means that the two clusters are most similar and have minimum pairwise distance. It repeats the last step until all data points have been merged. A *divisive* algorithm, on the other hand, starts with one cluster which contains all data points. Then, it separates

the cluster into two clusters using the weakest link between data points in the cluster. It repeats this step until each point is separated into its own cluster.

There are different methods to measure the distance between two clusters $\mathcal{A}$ and $\mathcal{B}$. The most common methods are:

- Single linkage: The single linkage distance between clusters $\mathcal{A}$ and $\mathcal{B}$ is defined by the minimum pairwise distance between each pair of points from the different clusters:

$$\min \{\text{dist}\,(x, y)\,,\ x \in \mathcal{A},\ y \in \mathcal{B}\}$$

- Complete linkage: The complete linkage distance between clusters $\mathcal{A}$ and $\mathcal{B}$ is defined by the maximum pairwise distance between each pair of points from the different clusters:

$$\max \{\text{dist}\,(x, y)\,,\ x \in \mathcal{A},\ y \in \mathcal{B}\}$$

- Average linkage: The average linkage distance between clusters $\mathcal{A}$ and $\mathcal{B}$ is defined by the average of all pairwise distances between each pair of points from the different clusters:

$$\frac{1}{|\mathcal{A}|\,|\mathcal{B}|} \sum_{x \in \mathcal{A}} \sum_{y \in \mathcal{B}} \text{dist}\,(x, y)$$

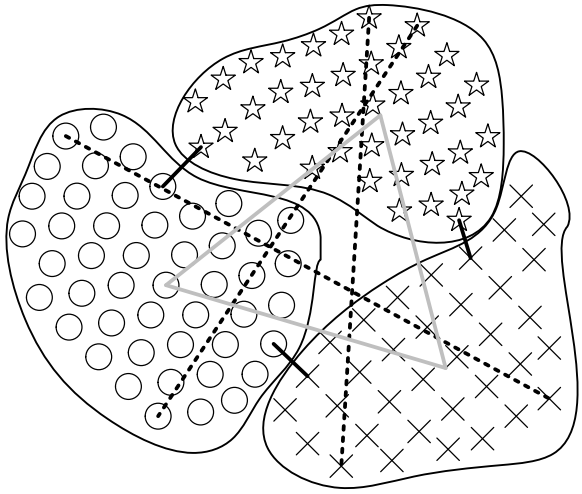

Figure 2.2.1: Different linkage types

Figure 2.2.1 shows an example of the three linkage types where the black solid lines represent the single linkage distances, the dashed lines represent the complete linkage dis-

tances, and the gray lines represent the average linkage distances between the given three clusters.

Partitional clustering The other type of clustering algorithm is called partitional clustering algorithms, which start by potential partitions or clusters of data points, then update these clusters iteratively using some objective function. The most well-known algorithm of this type is the k-means clustering.

***K*-means Clustering** The k-means algorithm is considered one of the best techniques of document clustering. It is also considered as a prototype type method for clustering, similar to nearest centroid classifier for classification. The prototype method's basic idea is to represent the input data by a set of points, prototypes, in the feature space. Usually the number of the prototypes is much less than the number of the input data points. In the case of the k-means, there is one prototype for each cluster which is the center point of the cluster. Each input data point is assigned to the cluster with the closest prototype.

The k-means algorithm starts with random centers for the clusters, which are called the clusters' centroids. Then, it assigns each data point to the nearest cluster centroid. For each cluster, it calculates the new cluster centroid by taking the mean value for all cluster members. It repeats the last two steps for a certain number of iterations or until convergence (the cluster assignment ceases to change).

There are different variants of the k-means algorithm such as k-medoids [22], fuzzy c-means [23], bisecting k-means [24], kernel k-means [25], and spherical k-means [26]. In spherical k-means, the distance measure that is used in clustering is the cosine distance, rather than the euclidean distance used in standard k-means. Spherical k-means is the most appropriate version of k-means for document clustering, as the documents are represented as vectors, and the most fit proximity measure used for the vectors is the cosine similarity. The main drawback of this algorithm is that its performance is highly affected by the initial centroids' values: therefore, there is no guarantee that it will converge to the global optimum. In the following section, we review some of the literature work that deals with the k-means initialization problem

Handling the *k*-means Cluster Center Initialization Problem: The study of selecting the appropriate values for the k-means initial class centers began a long time ago. One of the earliest works on the issue was by Forgy [27]. He suggest picking the initial cluster centers randomly from the input data points. In 1967, Macqueen [28] updated the k-means algorithm to work as follows: select the k initial centroids at random from the input data points as proposed by Forgy. Then assign the rest of the points to the nearest

cluster centroid. After each assignment, cluster centroids must be recalculated. Tou and Gonzales [29] proposed a simple cluster seeking algorithm (SCS). They suggest selecting the first centroid at random. Then, they sequentially pick the rest of the points one by one, and compute the distance between them and the initial cluster centroid. If the distance is greater than some threshold, they consider it to be the second cluster centroid. Then, they repeat the same steps to choose the k centroids. Kaufman and Rousseeuw [30] proposed an approach to pick the k initial centroids in a sequential manner. They select the most centered data point as the first initial centroid. Then, they select the second centroid which produces the lowest distortion according to some distortion function they defined in their work. The rest of the k centroids are picked in the same fashion. Pena et al. [31] have created a comparative study between the methods proposed by Forgy[27], Macqueen[28], and Kaufman and Rousseeuw[30].

The proposed approach of Katsavounidis et al.[32] starts by selecting a surface data point as an initial centroid. Then they search for the furthest data point from the initial centroid and set it as the second initial centroid. The distances from the rest of the points to the nearest seed are calculated. The point with the furthest distance from its nearest centroid is considered the next centroid. This process is repeated until they get the k initial centroids. Bradley and Fayyad [33] proposed another technique to initialize the cluster centers. They split the input data set into n subset(s). Then, they run the k-means clustering on each subset using the initial values as described by Forgy [27]. So, they end up with $n * k$ cluster centroids. Then they apply the k-means algorithm n time(s) on the whole data set, and the output centers of these n run(s) are used as the initial clusters' centroids.

Recently, Redmond and Heneghan[34] proposed a method that depends on the use of a kd-tree to perform a density estimation of the data at various locations. They apply a modified version of Katsavounidis' algorithm [32], which incorporates this density information, to choose k initial centroids. Suo et al. [35] have proposed a method of selecting the initial cluster centroids based on sub-graph division. They represent the entire set of input documents as the vertices of a graph based on the similarity matrix of the documents. Then, they connect the documents with distance less than some threshold to sub-graphs. They use the sub-graphs' centers as the initial centroids. In chapter 4, we will refer back to the k-means centroid initialization problem when we try to utilize the k-means algorithm to adapt the extracted background knowledge.

Spectral Clustering Spectral clustering is a family of clustering algorithms that use the spectrum, eigenvalues, of data to perform dimensionality reduction, then applying a clustering technique on it in the lower-dimension space. The similarity matrix between

each pair of data points is used to evaluate the eigenvalues decomposition. It starts by evaluating the similarity matrix S between each pair of data points in the input data set. In case of document clustering, dot product can be used to calculate the similarity matrix. Then, it constructs a graph whose vertices are the input data points, and whose edges are defined by the similarity matrix, S. It uses S to calculate the graph Laplacian matrix L, then solves the eigenvalues decomposition of the Laplacian matrix. The k eigenvectors that correspond to the smallest eigenvalues are used to represent the data in k-dimensional space where k is the required number of clusters. Then another clustering algorithm, typically k-means, is used to cluster the data.

There are several other approaches for calculating the similarity between data points. Some of these approaches use the local neighborhood of data points [36] instead of full pairwise similarity. An example of such an approach is the Gaussian similarity function: $S = exp\left(-d_{ij}^2/2\sigma\right)$ where $d_{ij}$is the distance between data point i and data point j, and σ is a parameter that controls the size of the neighborhood. Also, there are different methods of calculating the Laplacian matrix L from the similarity matrix. The standard, non-normalized, approach defines L as: $L = D - S$ where D is a diagonal matrix which is called the degree matrix, whose elements are the sum of similarity matrix where: $D_{ii} = \sum_{j=1}^{n} S_{ij}$. Another approach was presented by Meila and Shi [37] by normalizing L as follows: $L = I - D^{-1}S$. Ng et al. [38] proposed the use of another form of normalization for Laplacian matrix, $L = I - D^{-1/2}SD^{-1/2}$.

Non-negative Matrix Factorization (NMF) Clustering Non-negative matrix factorization is a family of algorithms that tries to factor a matrix X into two matrices Y and Z in such a way that the matrix X can be approximately reconstructed from the product of Y and Z as follows:

$$X \simeq YZ$$

There is also a constraint on the factor matrices Y and Z that both must be non-negative, i.e., all elements must be equal to or greater than zero. This approach of factorization was proposed before in other fields with different names. It was introduced to the machine learning field with this name by Lee and Seung [39, 40]. They investigated its properties and some useful algorithms for two types of factorizations.

Xu et al. [41, 42] have applied the NMF for document clustering. They suggested an algorithm for document clustering that factors a document term matrix X as a linear combination of latent concepts (or clusters): $X_{:j} = \sum_{\forall i} Y_{ij} Z_{:j}$ where Z' represents the latent concepts matrix. The algorithm first finds the non-negative factorization of the term-

document matrix X into Y and Z using the method proposed by Lee and Seung [39, 40]. Then documents are assigned to clusters according to their weights along the factors using equation $l_i = \arg\max_i Y_{ij}$.

2.2.2.3 Document Topic Indexing

Document topic indexing, or as it is sometimes called, document topic identification, is usually used to refer to the task of finding relevant topics for a set of input documents[3, 4]. It is used in many different real applications, such as improving retrieval of library documents pertaining to a specific topic. It could also be used to improve the relevancy of search engine results, by categorizing the search results according to their general topic and giving users the ability to choose the domain which is more relevant to their needs. There are some different tasks in text mining that fall under document indexing, including document tagging and keyphrase extraction. Medelyan [43] has classified these tasks according to two aspects. First, the source of the terminology that the topics are extracted from. Second, the number of topics that can be assigned to the documents. She has set three different values for the first aspect, which are: vocabulary-restricted, document-restricted, and no restriction. In vocabulary-restricted, the source of the topic is usually some sort of background knowledge such as a thesaurus or a structured glossary. In document-restricted, the source of the topics is the input documents themselves, where we try to select the most representative terms from these documents. In no-restriction, the topics that are assigned to the documents are selected freely with no restriction to a knowledge source.

The number of the topics aspect has three different values, which are: very few (main topics), detailed topics, and all possible topics. In main topics, the number of topics is limited to a small set of topics (usually less than a hundred). In detailed topics, more specific topics are included, which makes the number of topics much bigger (usually from hundreds to thousands), and usually more than one topic is assigned to each document. In all possible topics, the number of topics is limited to all the terms found in the input documents. This type is usually called full-text indexing.

In order to complete this classification of tasks, we would add a third aspect to these two aspects, which is the learning paradigm. As is well-known, learning paradigms can be classified into three types: supervised learning, semi-supervised learning, and unsupervised learning. Table 2.1 shows a list of possible topic indexing tasks classified based on these three aspects.

Considering document classification as a type of topic indexing is debatable, where the first task is to assign documents to classes, while the second one is to assign a topic to

Task	Source of terminology	Number of topics	Learning paradigm
Document classification	vocabulary-restricted	main topics only	supervised
Document clustering with cluster labeling	document-restricted	main topics only	unsupervised
Term assignment	vocabulary-restricted	detailed topics	supervised/ unsupervised
Keyphrase extraction	document-restricted	detailed topics	supervised/ unsupervised
Document tagging	unrestricted	detailed topics	supervised/ unsupervised

Table 2.1: Topic indexing tasks

each document. Lancaster has argued that this distinction is not fruitful [44]. We do not totally agree with the previous statement; we discuss this issue in more details in section 4.3.2.

We already discussed the details of document classification and document clustering in section 2.2.2.1 and section 2.2.2.2 respectively. There is a complementary task to document clustering, in order to be a complete topic indexing task; this task is called cluster labeling. Cluster labeling [45] is the process of selecting a representative label (topic) for each cluster obtained from the document clustering process. The source of these labels is usually from the terms and/or the phrases which the input documents are indexed with. One of the approaches used for cluster labeling is to use a feature selection technique, such as mutual information and chi-squared feature selection, to differentiate cluster labeling.

Term assignment, or subject indexing, is the process of finding the best representative topics for each document. The source of terminology is usually extracted from an external thesaurus, unlike the keyphrase extraction task where the main goal is to extract the most distinct phrases appearing in the documents. Lastly in document tagging, or in short tagging, tags can be chosen freely without any formal guideline. Usually the last three tasks, term assignment, keyphrase extraction, and tagging, are referred to as document topic identification. Although they differ in the source of terminology, they more or less do the same task, which is assigning each document a set of representative terms/phrases/tags.

Also, these three tasks have implementations for both learning paradigms.

The document topic identification task using background knowledge has been tackled in different ways in the literature. Coursey et al. [3, 46] proposed an unsupervised method based on a biased graph centrality algorithm, applied to a large knowledge graph built from Wikipedia. They map the input documents to Wikipedia articles based on the similarity between them, then they use their proposed biased graph centrality algorithm to find the matching topics. Similar is the work presented by Schönhofen in [47], but instead of using the Wikipedia full articles' contents, they used the articles' titles to match the input documents to the Wikipedia categories. Huynh et al. [48] suggested an update to the work proposed by Schönhofen in [47]: they added use of the hyperlinks in Wikipedia to use of the articles' titles, to improve topic identification.

Janik and Kochut [49, 50] have used the Wikipedia RDF (which is defined in [51]) to create their ontology. They then transfer the document text into a graph structure, employing entity matching and relationship identification. The categorization is based on measuring the semantic similarity between the created graph and the categories defined in their ontology.

This concludes the review of different text mining approaches for deriving patterns and trends that are related to our proposed work. Next, we review the different performance measures that we use to evaluate the performance of the proposed approach.

2.2.3 Performance Measures

Generally, text mining has two different sets of performance measures which depend on the learning paradigm of the technique under test: internal and external performance measures. As our proposed approach in chapter 4 is a semi-supervised approach, we will investigate only the external performance measures.

The external performance measures can be used for both supervised and unsupervised techniques. They are that set of measures which are concerned with measuring how the predicted data partitioning, either classes or clusters, is equivalent to a predefined (ground-truth) partitioning. In text mining, normally, this ground-truth partitioning is defined by human annotators and it is based on the semantics (the meaning) of the text. In this section, we refer to the ground-truth partitioning as categories.

Let $D = \{d_1, d_2, ..., d_m\}$ be the set input documents, let $\Phi = \{\phi_1, \phi_2, ..., \phi_l\}$ be the set of categories, and let $\Psi = \{\psi_1, \psi_2, ..., \psi_u\}$ be the set of clusters/classes obtained by a clustering/classification algorithm. In order to calculate the external measures, we need first to create the contingency matrix J, which shows the number of matches between each

cluster/class and each category. Hence, J can be represented as a matrix whose size is $l \times u$. If u is less than l, this means that the number of obtained clusters/classes is less than the expected number of categories. In this case, we augment the matrix J with zeros in such a way that the size of J becomes $l \times l$. Each element in the contingency matrix J_{ij} represents the number of the documents that are in category i and are clustered/classified as cluster/class j.

As the cluster labels are not known in document clustering output, the cluster-category mapping should be defined first, to be able to calculate most of the external measures. There are different methods for finding this mapping; they are used to reorder the contingency. We applied in this thesis the Hungarian algorithm [52] for the assignment problem, using the negation of the contingency matrix J as the input of the algorithm. We sorted the contingency matrix using the returned mapping in such a way that the category ϕ_i is mapped to the cluster ψ_i. Hence, the contingency matrix for clustering becomes very similar to the classification contingency matrix, where the elements on the diagonal of the matrix J_{ii} represent the number of correctly clustered/classified documents.

Most of the external performance measures are calculated based on defining four quantities that can be extracted from the contingency matrix for each category:

- TP_i (true positive with the category ϕ_i) is the number of documents which *belong* to category ϕ_i and are *correctly* clustered/classified under the cluster/class ψ_i. Using the contingency matrix, TP_i can be defined as $\mathrm{TP}_i = J_{ii}$.

- FP_i (false positive with the category ϕ_i) is the number of documents which *belong* to category ϕ_i but are *incorrectly* clustered/classified under a different cluster/class other than the cluster/class ψ_i. Using the contingency matrix, FP_i can be defined as $\mathrm{FP}_i = \sum_{j=1}^{l} J_{ji} - J_{ii}$.

- FN_i (false negative with the category ϕ_i) is the number of documents which *do not belong* to category ϕ_i and yet are *incorrectly* clustered/classified under the cluster/class ψ_i. Using the contingency matrix, FN_i can be defined as $\mathrm{FN}_i = \sum_{j=1}^{l} J_{ij} - J_{ii}$.

- TN_i (true negative with the category ϕ_i) is the number of documents which *do not belong* to category ϕ_i and are *correctly* clustered/classified under a different cluster/class other than the cluster/class ψ_i. Using the contingency matrix, TN_i can be defined as the set of documents that do not belong to TP_i, FP_i, nor FN_i. Hence we can define TN_i as $\mathrm{TN}_i = m - \mathrm{TP}_i - \mathrm{FP}_i - \mathrm{FN}_i$.

There are many different external measures based on the above quantities. We describe here some of the well-known external measures used in text mining. Some of these measures are used for document clustering, some are used for document classification, and some can be used for both.

- **Accuracy:** This measure is used for document classification. It measures how accurate the obtained classes are. It can be defined as the fraction of the documents which were correctly classified. Using the contingency matrix J defined above, the accuracy A_Ψ of a classification output can be calculated as follows:

$$\mathrm{A}_\Psi = \frac{\sum_{i=1}^{l} J_{ii}}{m}$$

  The higher the value of the accuracy, the more accurate the algorithm output obtained is.

- **Precision:** This measure can be used for both classification and clustering. Precision can be defined as the fraction of the documents that are clustered/classified to cluster/class ψ_i, and which belong to the category ϕ_i. Using this definition and using the defined contingency matrix J, we can calculate the precision P_i for a cluster/class ψ_i as follows:

$$\begin{aligned} \mathrm{P}_i &= \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i} \\ &= \frac{J_{ii}}{J_{ii} + \sum_{j=1}^{l} J_{ji} - J_{ii}} \\ &= \frac{J_{ii}}{\sum_{j=1}^{l} J_{ji}} \end{aligned}$$

  where $\sum_{j=1}^{l} J_{ji}$ is the total number of documents that are clustered/classified as cluster/class ψ_i. Then the overall precision P_Ψ of a clustering/classification output can

be calculated by averaging[1] P_i of all clusters/classes. The higher the value of the precision, the more precise the algorithm output obtained is.

- **Recall:** This measure can be used for both classification and clustering (it is also called the *sensitivity* measure in classification). Recall can be defined as the fraction of the documents of a category ϕ_i which are clustered/classified as cluster/class ψ_i. Using this definition and using the defined contingency matrix J, we can calculate the recall R_i for a cluster/class ψ_i as follows:

$$\begin{aligned} R_i &= \frac{TP_i}{TP_i + FN_i} \\ &= \frac{J_{ii}}{J_{ii} + \sum_{j=1}^{l} J_{ij} - J_{ii}} \\ &= \frac{J_{ii}}{\sum_{j=1}^{l} J_{ij}} \end{aligned}$$

where $\sum_{j=1}^{l} J_{ij}$ is the total number of documents that belong to the category ϕ_i. Similar to precision calculation, the overall recall R_Ψ of a clustering/classification output can be calculated by averaging[1] R_i of all clusters/classes. The higher the value of the recall, the better the algorithm output obtained is.

- **F-measure:** This is also called F-score or F_1score. It can be used for both classification and clustering, and was introduced by Rijsbergen [53]. It measures the output accuracy with regard to a specific ground-truth. It is defined as the *harmonic mean* of the precision and recall. Using the previous definitions of precision and recall, we can calculate the F-measure F_i for a cluster/class ψ_i as follows:

[1] This can also be calculated as a weighted average with respect to category sizes.

$$
\begin{aligned}
\mathrm{F}_i &= \frac{2\mathrm{P}_i\mathrm{R}_i}{\mathrm{P}_i + \mathrm{R}_i} \\
&= \frac{2\frac{\mathrm{TP}_i}{\mathrm{TP}_i+\mathrm{FP}_i}\frac{\mathrm{TP}_i}{\mathrm{TP}_i+\mathrm{FN}_i}}{\frac{\mathrm{TP}_i}{\mathrm{TP}_i+\mathrm{FP}_i} + \frac{\mathrm{TP}_i}{\mathrm{TP}_i+\mathrm{FN}_i}} \\
&= \frac{2\mathrm{TP}_i}{2\mathrm{TP}_i + \mathrm{FN}_i + \mathrm{FP}_i} \\
&= \frac{2J_{ii}}{\sum_{j=1}^{l} J_{ij} + \sum_{j=1}^{l} J_{ji}}
\end{aligned}
$$

The overall F-measure F_Ψ of a clustering/classification output can be calculated by averaging[1] F_i of all clusters/classes. The higher the value of the F-measure, the better the algorithm output obtained is.

- **Purity:** This measure is used for document clustering. It measures how pure each cluster is. It is defined as the ratio between the dominant category ϕ_j in the cluster ψ_i, and the size of this cluster. Hence, the purity Pu_i of the cluster i can be evaluated as follows:

$$
\mathrm{Pu}_i = \frac{\max\limits_{\forall j}(J_{ji})}{\sum_{j=1}^{l} J_{ji}}
$$

Note that, $\max\limits_{\forall j}(J_{ji})$ is not always equivalent to J_{ii}, as sometimes the dominant category in the cluster ψ_i is not the category ϕ_i. The total purity for the clustering Pu_Ψ can be defined as the weighted average of the purity of all clusters, which can be formulated as follows:

$$\begin{aligned}
\mathrm{Pu}_{\Psi} &= \sum_{i=1}^{u} \frac{m_i}{m} \mathrm{Pu}_i \\
&= \sum_{i=1}^{u} \frac{\sum_{j=1}^{l} J_{ji} \max_{\forall j} (J_{ji})}{m \sum_{j=1}^{l} J_{ji}} \\
&= \frac{1}{m} \sum_{i=1}^{u} \max_{\forall j} (J_{ji})
\end{aligned}$$

Note that, if $\max_{\forall j} (J_{ji})$ is equivalent to J_{ii} then the purity measure is similar to the accuracy measure for the classification. The higher the value of the purity, the better the obtained output is.

- **Entropy:** Entropy is one of the common external measures that is used for document clustering. It evaluates the homogeneity of the clusters with respect to the ground-truth categories. The higher the homogeneity of the clusters, the lower value of entropy is obtained, hence, the better are the clusters. To evaluate the entropy of the clusters, we need to calculate for each cluster ψ_j the probability that the members of that cluster belong to a specific category ϕ_i, $\mathbb{P}_{ij} = \frac{J_{ij}}{\sum_{i=1}^{l} J_{ij}}$. Hence, the entropy of the cluster j can be calculated as follows:

$$\begin{aligned}
\mathrm{E}_j &= -\sum_{i=1}^{l} \mathbb{P}_{ij} \log (\mathbb{P}_{ij}) \\
&= -\sum_{i=1}^{l} \frac{J_{ij}}{m_j} \log \left(\frac{J_{ij}}{m_j} \right)
\end{aligned}$$

where $m_j = \sum_{i=1}^{l} J_{ij}$ is the number of the documents in cluster ψ_j. The total entropy for the whole output E_{Ψ} can be defined as the weighted average of the entropy of all clusters, which can be formulated as follows:

$$\begin{aligned} \mathrm{E}_{\Psi} &= \sum_{j=1}^{u} \frac{m_j}{m} \mathrm{E}_j \\ &= \frac{-1}{m} \sum_{j=1}^{u} \sum_{i=1}^{l} J_{ij} \log \left(\frac{J_{ij}}{m_j} \right) \end{aligned}$$

The lower the value of the entropy, the better the obtained output is.

- **NMI:** Normalized Mutual Information (NMI) is a well-known document clustering performance measure. It estimates the amount of shared information between the clusters' labels and the categories' labels. It measures the amount of information that can be obtained from the cluster labels by observing the category labels. It can be written as a function of the ground-truth partitioning Φ and the output cluster partitioning Ψ as follows:

$$\mathrm{NMI}\left(\Phi, \Psi\right) = \frac{\mathrm{MI}\left(\Phi, \Psi\right)}{\sqrt{\mathrm{H}\left(\Phi\right) \mathrm{H}\left(\Psi\right)}}$$

where $\mathrm{MI}\left(\Phi, \Psi\right)$ is the mutual information between ground-truth Φ and the output clusters Ψ, and $\mathrm{H}\left(\Phi\right)$ and $\mathrm{H}\left(\Psi\right)$ are the entropies of the ground-truth Φ and the output clusters Ψ respectively. $\mathrm{MI}\left(\Phi, \Psi\right)$ can be defined as:

$$\begin{aligned} \mathrm{MI}\left(\Phi, \Psi\right) &= \sum_{i=1}^{l} \sum_{j=1}^{u} \mathbb{P}\left(\phi_i \cap \psi_j\right) \log \frac{\mathbb{P}\left(\phi_i \cap \psi_j\right)}{\mathbb{P}\left(\phi_i\right) \mathbb{P}\left(\psi_j\right)} \\ &= \sum_{i=1}^{l} \sum_{j=1}^{u} \frac{J_{ij}}{m} \log \frac{m J_{ij}}{m_i m_j} \end{aligned}$$

where $\mathbb{P}\left(\psi_j\right)$, $\mathbb{P}\left(\phi_i\right)$, and $\mathbb{P}\left(\phi_i \cap \psi_j\right)$ are the probabilities of a document being in cluster ψ_j, category ϕ_i, and in the intersection of ψ_j and ϕ_i respectively, $m_j = \sum_{i=1}^{l} J_{ij}$ is the number of the documents in cluster ψ_j, and $m_i = \sum_{j=1}^{u} J_{ij}$ is the number of the documents in category ϕ_i. The entropies $\mathrm{H}\left(\Phi\right)$ and $\mathrm{H}\left(\Psi\right)$ measure the amount of information contained in the cluster and category labels respectively. They can be

calculated as follows:

$$\begin{aligned} \mathrm{H}(\Phi) &= -\sum_{i=1}^{l} \mathbb{P}(\phi_i) \log \mathbb{P}(\phi_i) \\ &= -\sum_{i=1}^{l} \frac{m_i}{m} \log \frac{m_i}{m} \end{aligned}$$

$\mathrm{H}(\Psi)$ can be evaluated similarly. The higher the value of the NMI, the better the obtained output is.

2.3 Ontology

Due to the rapid growth of the Internet, the availability of different knowledge sources has been increased dramatically. Many of these knowledge sources are structured in the form of a knowledge repository, such as web directories, wikis, online encyclopedias, and so on. Recently, a new line of research introduced the use of knowledge repositories to enhance the efficiency of different text mining tasks. They start first by extracting the background knowledge from one of these knowledge repositories in such a way that it can be used in their presented application in text mining. Usually, they structure this background knowledge in the form of an ontology. Although most of the works done in this field do not mention explicitly that they are creating an ontology, the form of the background knowledge that they use can be considered as a form of a lightweight ontology, as we will see later. So, we will start by defining what is meant by ontology, then review some of the work done which has used ontologies in different text mining tasks.

The word ontology has been applied in many different ways according to the domain that it was used in. It is originally defined in philosophy as a systematic account of existence. Then, it was borrowed in AI to express what exists as what you can represent [54].

Gruber has defined it as "an explicit specification of a conceptualization" in [55, 54]. A conceptualization here refers to an abstract form of human thoughts about different things in the world. An explicit specification means that the concepts in that conceptualization, and their relations and properties, must be explicitly named and defined. Guarino and Giaretta [56] have altered this definition slightly to "an explicit account or representation of some part of a conceptualization" to be more generic, as they tried to soften the definition of an ontology to cover only part of a conceptualization. Garshol [57] has given a clearer

definition for ontology within computer science as "a model for describing the world, that consists of a set of types (concepts), properties, and relationship types". In other words, an ontology defines different kinds of things in some conceptualization. These kinds of things are called classes (or concepts). These concepts have some properties (also called slots) that describe the features and the attributes of them. These properties have some restrictions (sometimes called facets) on the values that they can take. The concepts are arranged in the form of a taxonomy of concepts and sub-concepts, which means that this taxonomy defines the explicit relations between the defined concepts. We can see that there is some similarity between ontology development, in this manner, and object-oriented design process. Despite this, there are also some significant differences between them[2].

The major difference is that object-oriented design emphasizes the operational properties of a class, whereas ontology design decisions are based on the structural properties of a class. Consequently, a class description and properties, and relations among classes in an ontology, should be different from the structure for a similar domain in an object-oriented design [58]. Noy et al. [59] showed that the process of developing an ontology consists of the following four steps:

1. Defining classes in the ontology.
2. Arranging the classes in a taxonomic (subclass–super-class) hierarchy.
3. Defining classes' properties and describing their restrictions.
4. Filling in the values for properties for instances.

2.3.1 Lightweight Ontology

To this point, it is still debatable what can be considered as an ontology and what cannot. Nevertheless, most ontologies have at least:

- a list of terms that refers to the concepts.
- some definitions of the semantic relations between these concepts.

There are different types of ontology used in different types of application. They vary in terms of usability from human-readable ontology, such as the Semantic Wiki [60], to

[2]A detailed comparison between object-oriented and ontology implementation issues is discussed in http://www.w3.org/TR/sw-oosd-primer/#comparison

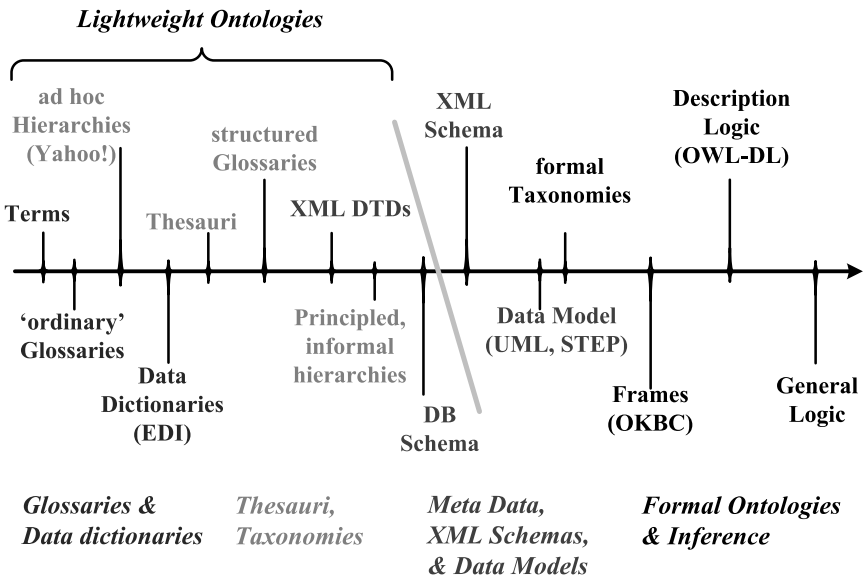

Figure 2.3.1: Kinds of ontologies. Adopted from [1]

an ontology which can only be processed by machine. In terms of coverage, they vary from upper-level generic ontologies [61] to domain-specific ontologies, such as a biological ontology [62]. They also vary in the degree of expressiveness, from lightweight ontology to formal ontology. An ontology that consists of a list of terms and little specification of the semantic relations between these terms is considered a lightweight ontology (LWO). A taxonomy can be considered a good example of a LWO. On the other hand, a full fledged ontology (formal ontology) contains more expressive concepts and relations. Its concepts are represented in more detail by including more description of the concepts, their properties, and restrictions of the values for these properties, going even further by including additional semantic relationships between the concepts other than the simple taxonomy (is-a) relationship. Figure 2.3.1 shows a continuum of kinds of ontologies. It shows that ontologies can vary from LWO, such as ordinary glossaries and thesauri, to more formal ontologies with inference, such as frames and description logics. On the other hand, what should not be considered an ontology is any representation of concepts without a clear explicit definition of them. Latent Semantic Analysis (LSA) is a good example of what should not be considered an ontology since it is, as its name implies, a representation of knowledge in the form of a list of semantically related latent concepts, which mean that there is no explicit definition for these concepts.

Ontology Examples

The following is a list of some famous and large scale ontologies used in different domains:

- ***WordNet***: a lexical database for the English language [63].
- ***Basic Formal Ontology***: a formal upper-level ontology [64].
- ***BMO***: an e-Business Model Ontology based on a review of enterprise ontologies and business model literature [65].
- ***Cyc***: an artificial intelligence project that attempts to assemble a comprehensive ontology and knowledge base of everyday commonsense knowledge, with the goal of enabling AI applications to perform human-like reasoning [66].
- ***OBO Foundry***: a collaborative experiment involving developers of science-based ontologies. The foundry is concerned with establishing a set of principles for ontology development, with the goal of creating a suite of orthogonal interoperable reference ontologies in the biomedical domain [67].

2.3.2 Ontology Evaluation

As we mentioned, ontologies vary in usability, coverage, and the degree of expressiveness. Hence, there may exist many different ontologies for the same conceptualization of a specific domain of knowledge. We should be able to determine which of the present ontologies can fit our requirements. Thus, there is a need for a definition of some ontology evaluation methods, to help in selecting the best matching ontology for any task or application. Also, ontology engineers need to have a method to judge their created ontologies and possibly help them to adjust them. As well, for the automated and semi-automated ontology learning techniques, ontology evaluation is needed to tune the ontology parameters and values.

There are different evaluation approaches that are present in the literature. Brank et al. [68] have classified them into four different categories of evaluation techniques, based on the kind of the ontologies that are being evaluated and for what purpose they are used, as follows:

- **Golden standard evaluation**: Comparing the ontology to a "golden standard" [69]. In this approach, the created ontology is compared with a manually created ontology which is built by domain experts.

- **Application-based evaluation**: Using the ontology in an application and evaluating the results [70].
- **Data-driven evaluation**: Comparisons with a source of data about the domain to be covered by the ontology [71]. This form of data is usually in the form of a collection of text documents, web pages, or dictionaries.
- **Assessment by humans evaluation:** Human evaluation is the most popular evaluation method. The evaluation is done by humans who try to assess how well the ontology meets a set of predefined criteria, standards, requirements, etc. [72].

2.3.3 Using Ontologies in Text Mining

In text mining and information retrieval, ontologies have been employed in many different tasks such as query expansion and document representation. In both tasks, an ontology is used to better represent text semantics, in order to retrieve documents which are more relevant to the input query. A possible application of ontology in text mining is the enrichment of document modeling. In this application, instead of using the conventional "bags of words" model, one can model the document using concepts. And, rather than using the simple structure of "bags", we can use a more sophisticated structure such as the one found in ontologies. Thus, documents will be represented in a structure of concepts that reflects the document meaning, rather than using a collection of words that just occurred in the document. The process of representing the documents in concept space has many names including semantic indexing [73], concept mining, and explicit semantic analysis [74]. Nevertheless, most of the recent works in the literature that utilize an ontology to improve a text mining task do not use the term "ontology" to describe their technique; rather, they use the term "background knowledge" or even the name of the ontology itself, such as WordNet. Most of these techniques, as we review in the following subsections, use a form of lightweight ontology.

Ontology Applications in Document Representations In the remaining part of this section, we explore some of the work done in the literature which uses different types of ontologies to give a better representation of documents. These representations will be used to improve the performance of different text mining applications.

Aussenac and Mothe [75] used ontologies as background knowledge to explore a collection of documents. First, they built up a domain-specific ontology from a collection of text documents using tools like KAON and Terminae. KAON [76] is is a standard workbench

for the analysis of texts in German and English. Terminae is a general workbench to build up ontologies from large text corpora in French or English [77]. Then, they constructed a hierarchy on top of the ontology with the help of the ontology entries. Each hierarchy corresponds to a dimension on which documents can be mapped. Next, they associated documents to the concepts from the different concept hierarchies using an approach that they proposed in [78]. Finally, they built up a tool that helped the user to navigate and explore the documents using the hierarchy they have defined. They did not supply any evaluation measures for their technique.

Shehata et al. [79] proposed a new method to build an ontology using the set of the documents to be processed. They presented a new concept-based model to be used for document representation, wherein they extract concepts based on terms' relationships at both sentence and document levels. The concept-based model can effectively discriminate between non-important terms, with respect to sentence semantics, and terms which hold the concepts that represent the sentence's meaning. The term which contributes to the sentence semantics is assigned two different weights to introduce their importance in both sentence and document levels. These two weights are combined into a new weight. The concepts that have maximum combined weights are selected by the concept extractor.

Choudhary and Bhattacharyya [80] proposed a new method for constructing the feature vector that represents a document in a document clustering task. They use the Universal Networking Language (UNL) to determine the universal words of the documents (UWs) and the importance of each UW in the document. The UNL is an artificial language that can be used as a knowledge representation language in information retrieval applications. It is organized in an ontology-like structure based on language sentences. It is described in detail in [81]. They proposed two different techniques for document representation. In the first technique, they used the UNL Graph Links to determine the importance of each UW. They assigned to each UW in the document a weight, which represents the importance of that UW for the document. This was calculated using the number of links that connected this UW to other UWs in the document. They updated these weights in the second technique, by weighting each link label in the graph so that the weight of each UW equals to the summation of all links' weights which connected this UW to other UWs in the document. They showed that by using the first technique, they achieved better improvement (about 16%) in document clustering than the basic BOW technique. Further, they achieved even greater improvement (about 26%) by using the second technique.

WordNet-Based Approaches

WordNet is one of the most famous lexical databases for the English language. It defines the type of each English word, whether it is a noun, verb, adverb, or adjective. It groups the words which are synonymous with one another into synonym sets, named synsets. Each synset represents one underlying lexical concept. Each word may have numerous different senses, which means that it would belong to various synsets. For example, a synset which is described by "A motor vehicle with four wheels; usually propelled by an internal combustion engine" represents a concept which contains the words *car, auto, automobile, motorcar, and machine*. The synsets also have different semantic relations with each other, like antonyms, hypernyms, hyponyms, coordinates, holonyms, and meronyms. These synsets and the relations between them define the WordNet linguistic ontology.

The WordNet ontology was developed manually by the Cognitive Science Laboratory at Princeton University under the direction of George A. Miller [82, 83]. WordNet has been applied to a variety of problems in machine learning, natural language processing, information retrieval, and artificial intelligence. In this section, we discuss some of the methods which used WordNet as an ontology for document representation.

Hotho et al. used WordNet to improve text document clustering [84, 85, 86]. They proposed different strategies to amend the "bag of words" document representation with concepts from WordNet. They have three different sets of strategies for extending the bag of words model. These sets of strategies are generic and can be applied with other ontologies, therefore we review their method in some detail. The first set of strategies deals with how to represent the concepts in the term vector model. They suggest three different strategies: either concatenating the concepts with the term vectors, replacing the terms that have representative concepts with these concepts, or using the representative concepts only.

The second set of strategies deals with the disambiguation of terms. Two different disambiguation strategies are proposed: first concept and context concepts strategies. In the first concept strategy, they used the first concept in the ordered list of concepts returned from WordNet, since it represents the most common representative concept in the synset. In the context concepts strategy, they used a simplified version of word sense disambiguation defined by Agirre et al. [87].

The last set of strategies is concerned with the hypernyms of the concepts. In this strategy, they try to assign to generic concepts higher weights than the specific concepts. They add to each concept weight the sum of the weights of its sub-concepts that occur in the same document up to a given n level. So that, $n = 0$ means that they add nothing to the weight of the concept and $n = \infty$ means that they add all the weights of the concept's

sub-concepts. The cluster algorithm that they have used is the bisecting k-means[24]. They have applied their approach to two data sets: Reuters-21578[3] [88] and Java-Corpus data set.

They reported the results with different performance measures: purity, inverse-purity, f-measure, and entropy. Different observations have been reported out of their work. The best strategy for representing the concepts is by augmenting the concepts vector to the term vectors. The best disambiguation strategy is the context strategy, since the "first concept" strategy does not improve the results significantly. One last conclusion they have drawn is that the best number of hypernyms levels is 5. They have achieved a significant result by using the above framework.

Sedding et al. [89] proposed an update to the approach described above by Hotho et al. They suggested using Part-of-Speech Tagging (PoS) as a new strategy for word disambiguation. They compared their results with the basic VSM. They observed that when using PoS only, the performance of the clustering does not change. They also noticed that the best number of hypernyms levels is 5, and the performance is decreased dramatically when using all hypernyms levels ($n = \infty$).

One of the recent researches that used WordNet semantics for document clustering was done by Gad and Kamel [90]. They have proposed an update to the standard term weighting function (tf-idf) using the semantically related terms in the same document. The new weight is calculated as follows:

$$\tilde{w_{ij_1}} = w_{ij_1} + \sum_{j_2=1, j_2 \neq j_1}^{n} w_{ij_2} \times \mathrm{sim}_{Lesk}\left(t_{j_1}, t_{j_2}\right)$$

where w_{ij} is the tf-idf weight of term j in document i, and $\mathrm{sim}_{Lesk}\left(t_{j_1}, t_{j_2}\right)$ is the semantic information between terms j_1 and j_2 using the adapted Lesk algorithm proposed by Lesk [91].

2.3.4 Using Wikipedia in Text Mining

Wikipedia[4] is a free online encyclopedia. Its contents have been written collaboratively by a large number of volunteer contributors around the world. Wikipedia web-pages can be edited freely by any internet user. This leads to a rapid increase of its good-quality

[3] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[4] http://en.wikipedia.org/

contents, as any potential mistakes are quickly corrected within the collaborative environment. Wikipedia coverage of topics has become as comprehensive as other well-known encyclopedias such as Britannica[5] [92], with reasonable accuracy.

The strength of Wikipedia lies in its size and its wide coverage for different topics, which could be used to overcome the limitations of the coverage and scalability of other knowledge bases and ontologies. These features have encouraged research using Wikipedia to build a well-structured knowledge base which can be used in different text mining tasks. In the following sections, we discuss some of the applications proposed using Wikipedia as a knowledge base.

Computing Semantic Relatedness

One of the initial tasks that involves Wikipedia as a knowledge base is computing the semantic relatedness between documents (Wiki-Relate) [93]. They proposed to take the Wikipedia categorization system as a semantic network, which served as a basis for computing the semantic relatedness of words.

Gabrilovich and Markovitch introduced the idea of representing the terms and the documents in the Wikipedia concept space to compute semantic relatedness between fragments of natural language text [74]. They regarded each Wikipedia Article as a concept. Each term is represented as a vector of weights in the direction of all concepts that this term occurred in. Similar to the Latent Semantic Analysis (LSA), which maps the document to a latent lower dimension space, they map them to the concept space. They named their approach "Explicit Semantic Analysis" (ESA), as they use concepts that are explicitly defined by users. They reported that the performance is significantly better than other state of the art methods.

After the idea of mapping the documents to the Wikipedia concepts space presented by Gabrilovich and Markovitch, many researchers have used this approach to accomplish different tasks. For example, Syed et al. [94] introduced a similar approach to build a Wikipedia ontology, which they named Wikitology. They consider each Wikipedia article as a specific concept as well, and each Wikipedia category as a generic concept. In their approach, they measure the pairwise distance between the input documents and all the Wikipedia articles using the conventional cosine distance. Then, they select the most similar Wikipedia articles for each document and set the title of these articles as the most specific concepts for that input document. After that, they identify the more generic concepts for each input document using Wikipedia inter-category article relationship. They showed the effectiveness of their approach using some preliminary results.

[5] http://www.britannica.com/

Wikipedia has been used also for text classification. Pu Wang et al. [95] have presented a new approach to improve text classification using Wikipedia as background knowledge. They built a large thesaurus from Wikipedia, and then used this thesaurus to build an ontology of concepts. They extracted the relationships between the concepts of the thesaurus from the structure of the Wikipedia's knowledge. For example, they extracted the synonymy relationships between concepts from the "Redirect" links, polysemy relationships from the "Disambiguation pages", and hyponymy from the category structure of the Wikipedia. By using this thesaurus, they overcame the problems of the original vector space model. They claim that by using their approach, text classification performance improved significantly with respect to the baseline algorithm (VSM). Other researches that use Wikipedia for text classification can be found in [96, 97, 98, 95].

Wikipedia has been used as an online knowledge base for conducting several other text mining researches. In [99, 100, 101, 102, 103], Wikipedia was used to enhance a text clustering task. It was also used for driving a large scale taxonomy in [104].

2.4 Summary

In this chapter, several text mining tasks have been reviewed. These tasks include document classification, document clustering, and document topic indexing. This revision was done in order to outline the features and properties of these tasks, as we are going to compare them with our proposed approach later. We also covered the definition of the ontology and reviewed different approaches which employ it in text mining. As we have used Wikipedia as a knowledge source to create the ontology later, we surveyed a portion of the research done in the literature which utilizes Wikipedia as a source of background knowledge to accomplish different text mining tasks.

Chapter 3

Extracting an Ontology from a Human Knowledge Repository

As we discussed in chapter 2, ontologies have been employed in a variety of tasks for text mining. These ontologies vary from very lightweight ontologies where the concepts are presented as a set of weighted terms [96, 94], to somewhat well-structured ontologies such as the WordNet. One of the first works that introduced the use of the co-occurrence theory to generate a lightweight ontology was done by Ding and Engels [105]. In this chapter, we introduce a new approach to create a lightweight ontology from a knowledge repository. We start by discussing the required features of a knowledge repository that can be used in our approach, then present the steps we followed to build our ontology. After that, we apply this approach to one of the well-known knowledge repositories, namely Wikipedia. We discuss the different features and issues associated with using Wikipedia as a knowledge repository. Lastly, we introduce the different applications that can use our created ontology in different text mining tasks.

3.1 Overview

Creation of an ontology consists of four steps as we described in section 2.3: defining the ontological concepts, defining the taxonomical relationships between the concepts, defining the properties and restrictions of the concepts, and filling the values for these properties. In our case, we are building a lightweight ontology which has a simpler structure than a full-fledged ontology. Our ontology consists only of concepts, the taxonomy connecting the concepts, and the concept-term relation. Concept-term relation represents the only

concept property in our ontology, mainly used for different text mining tasks. Also, it can be used to find the non-taxonomical relation between concepts, which we call the semantic relation. Thus, there are some minimum requirements that should be met in order to be able to create our ontology. In the following subsection, we discuss these requirements in detail.

Knowledge Repository Requirements

There are three main features that should be met by this Knowledge Repository (KR) to create our ontology:

1. The first feature is associated with the first step in ontology creation, which is the definition of ontological concepts. We need the KR to contain a collection of topics which covers a wide range of human knowledge. These topics must have explicit labels that represents the idea of the topic, such as "Biology" to represent a topic that covers biology-related concepts. We use these topics to represent the concepts of our ontology, which are considered the basic building blocks of the ontology. Formally, for a Knowledge Repository (KR), we can define the set $\mathcal{C}$ to represent the list of all ontological concepts as follows:

$$\mathcal{C} = \{c_i : \ i = 1, .., n\} \tag{3.1}$$

   where n is the total number of concepts in our ontology.

2. The topics, concepts, in the KR should have some taxonomical relationship to one another. These taxonomical relations should be in the form of a parent-sibling relationship. This relation may cover any of the following relations: part-of, inherited-from, instance-of (is-a), or any type of relation of that nature. Thus, when we move in the up direction of the relation, we move to more abstract or general topics. Conversely, when we move in the down direction of the relation, we move to the more specific topics. We call this relationship a hypernym. So, the hypernym relationship between two concepts can be defined by $\mathcal{P}$ where

$$\mathcal{P} = \{(c_i, c_j) : \ i = 1, .., n; \ j = 1, .., n; \ i \neq j\} \tag{3.2}$$

   shows that concept c_i is a hypernym for concept c_j.

3. The KR must have a huge set of textual data, articles, $\mathcal{A}$ which constitute the detailed knowledge of the KR. We use these articles to extract the background knowledge.

Each topic in the KR should be associated with a collection of these articles which describes, discusses, and presents the ideas of that topic. We use these textual data to construct the concept-term relation for each concept. Hence for each concept c_i, we can define the set of articles $\mathcal{A}_{c_i}$ as follows:

$$\mathcal{A}_{c_i} = \{a_j : \ j = 1, .., m_{c_i}\} \tag{3.3}$$

where m_{c_i} is the total number of articles that fall under the topic c_i.

Figure 3.1.1 shows an example of the required knowledge repository structure, where the upper topics represent more abstract ideas, and leaves of these topics represent more specific ideas. For example, for the topic "Biology" which represents a more generic topic, the topics "Anatomy", "Ecology" and "Zoology" are a subset of the leaves of that topic. Moreover, each topic (*concept*) has a list of related articles that expresses the ideas of this topic.

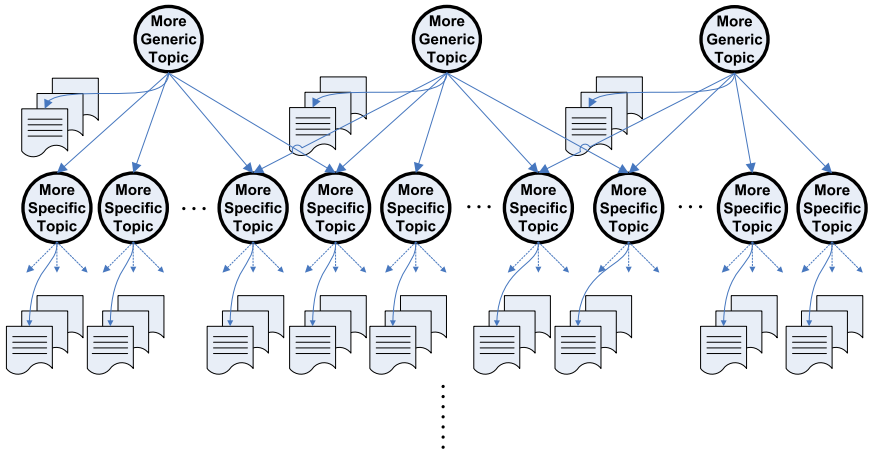

Figure 3.1.1: Hierarchical Knowledge Repository

Due to rapid advances in information processing, and Internet growth, many sources of world knowledge have become available in recent years. This kind of wide-ranging knowledge repository can be found in different web directories such as the Open Directory Project (ODP), Google Web Directory (which is based on the ODP Hierarchy), and Yahoo! Web Directory. Also it can be found in various types of Encyclopedias such as Wikipedia.

3.2 Ontology Creation Framework

In this section, we show how we build the ontology using any knowledge repository. In section 3.3, we point out some implementation issues which we have when using Wikipedia as the knowledge repository. This process of creating the ontology consists of three main modules: Extracting the Ontology Taxonomy, Manipulating Knowledge Repository Articles, and Building the Concept-Term Mapping. We discuss each module in detail in the following subsections.

Basic Idea

To build an ontology from a knowledge repository, we need first to extract the ontology tree from the knowledge structure. This step involves many different issues which will be dealt with in detail later in this chapter. Next, we extract the representative terms from the articles under each node of the tree constructed in the previous step. Last comes the concept-term mapping, completing the construction of the ontology.

After explaining the steps of building the ontology, we show how to use it in different text mining applications. Figure 3.2.1 outlines the proposed ontology creation framework.

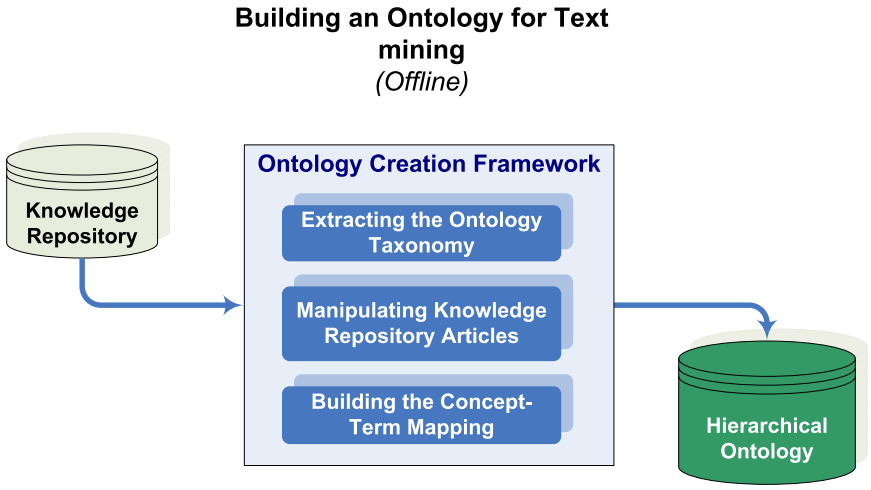

Figure 3.2.1: Ontology creation framework

As mentioned above, each topic in the KR represents a concept which has some articles associated with it. Using these articles, we want to extract the set of terms that represent these articles. After extracting these terms (discussed in detail later in this chapter), let $\mathcal{T}$ be defined as the set of all terms found in the KR:

$$\mathcal{T} = \{t_j : \ j = 1, .., l\}$$

where l is the total number of terms found in the KR. Then we use the concept-articles relation $\mathcal{A}_{c_i}$ defined in equation (3.3) to associate to each concept c_i a list of terms $\mathcal{T}_{c_i}$. Furthermore, we need to associate with each term a weight that represents the strength of the relation between this term and that concept. Thus, for each concept c_i, we need to define a list of term and weight pairs as follows:

$$\mathcal{W}_{c_i} = \{(t_j, w_{c_i j}) : \ j = 1, .., l\}$$

These weights are calculated based on the frequency of occurrence of these terms in the articles under that concept. In other words, the weight $w_{c_i 1}$ is calculated based on the frequency of occurrence of the term t_1 in the set of articles $\mathcal{A}_{c_i}$. Note that if a term t_j is not found in any of the articles A_{c_i}, the weight $w_{c_i j}$ is set to 0. So, we can define the mapping from all concepts to terms as $\mathcal{M}$, where $\mathcal{M}$ is defined as

$$\mathcal{M} = \{(c_i, \mathcal{W}_{c_i}) : \ i = 1, .., n\} \tag{3.4}$$

Thus, $\mathcal{M}$ can be defined as the mapping function which associates each concept $c_i \in \mathcal{C}$ to each term $t_j \in \mathcal{T}$. Our extracted ontology can be represented with the following three main components: the set of concepts, $\mathcal{C}$ (3.1); the taxonomical concept relations, $\mathcal{P}$ (3.2); and the concept-term mapping, $\mathcal{M}$ (3.4). This concludes the basic idea of the ontology extraction process. Next, we describe the steps of ontology creation in detail.

3.2.1 Extracting the Ontology Taxonomy

The first step in ontology creation is defining ontological concepts. As we mentioned above, a concept in the ontology must have a representative label, and it must cover a topic or an idea of the human knowledge. So for example the label "Biology" can represent a concept, while the label "Foreign" cannot. The "Foreign" label does not represent a concept as it is a very ambiguous term. But the label "Foreign policy" or "Foreign language" can represent topics. Usually, the process of filtering KR topics to decide what to include and what to not include in the ontology concept list is a KR-based issue, which differs from one KR to another. For example, the structure of the web directories can be used more or less

as-is, as when it was built it was meant to be used as a taxonomy of topics. On the other hand, the article or category structure of a KR like Wikipedia was not built to represent a taxonomy, so it was not taken into account that they may be used for tasks like ontology creation. Hence, we cannot use this structure of the knowledge repository as-is.

In the literature, many tools are available to create a taxonomy from a collection of documents, such as TaxGen [106]. Ponzetto and Strube [104] have proposed a Natural Language Processing (NLP) approach to construct a taxonomy from Wikipedia. They extract the generic *is-a* relations from Wikipedia category links. As this task of defining the ontology concepts is a KR-based issue, we discuss it in detail with the Wikipedia implementation issues (section 3.3).

After defining the set of concepts $\mathcal{C}$ as in equation (3.1), we need to organize the taxonomical relations between concepts in a form that can be used in later steps in text mining. A taxonomy can be represented as a tree or as a directed graph. In our case, the tree representation cannot be used as each concept may have multiple parent concepts, so we have used the directed graph representation to represent our ontology's taxonomy. In this graph, each node represents a concept, each edge (arc) represents a taxonomical relation between two concepts, and the direction of the edge maps the hypernym relation. In other words, if a concept c_i is a hypernym for a concept c_j, there will be a directed edge that goes from the node that represents the concept c_i to the node that represents the concept c_j.

One of the possible mathematical formulations of a graph is the adjacency matrix. An adjacency matrix, X, is a square matrix with size $n \times n$ where n is the number of the graph nodes (the number of concepts in our case). Hence, we can represent the taxonomic hierarchy of our ontology as an adjacency matrix H. If the entry H_{ij} of the adjacency matrix H equals one, this means that the concept c_i is a hypernym for a concept c_j.

One of the issues of the KR taxonomies is that they may contain cycles. A cycle is a path that starts from a concept and ends with the same concept. Taxonomy cycles should not exist in the ontology's taxonomy. The cycles can be a direct cycle (loop) where there exists an edge from a concept to itself. Handling direct cycles, loops, is very trivial. It can be done by setting the diagonal values of the matrix H to zero. Obviously, we cannot remove other larger cycles manually as the number of concepts is huge and the length of the path of a cycle can be very long. In other words, we want to convert the directed graph represented in H to a directed acyclic graph, DAG, automatically. This can be done by removing one of the edges, arcs, that exist in each cycle.

There are two possible methods to convert a directed graph to DAG based on the available information for the given taxonomy. The first method is the simpler one. We

assume that the top level concepts, the hierarchy roots, are given with the taxonomy. Algorithm 3.1 shows how to convert a graph to a DAG given that the root concepts are known. This algorithm is more like the breadth-first tree traversal. It starts by setting the root concepts in the open set, the nodes to be visited, and sets the closed set, the already visited nodes, to empty. Then, for each node in the open list, it check its fan-out links. If any of them points back to one of the already traversed nodes, this arc is removed. Next, it put this node in the closed set and removes it from the open set. Note that this algorithm does not try to minimize the number of removed arcs.

If the root concepts are not given, the problem becomes much harder. The assessment of which edge can be removed without dramatically affecting the taxonomy is hard to make. In other words, we want to select as few edges as possible to be removed from the taxonomy graph to convert it to a DAG. This problem is an NP-hard problem called the minimum feedback arc set problem, FAS [107, 108], or the maximum acyclic sub-graph problem [109]. There are many algorithms in the literature that try to use different heuristics to solve this problem [110]. One possible reduction of this problem is to find the best vertex sequence of all graph vertices in such a way that the leftward arcs, from a vertex to any of the vertices on the left hand side of it in the sequence, are minimized. Then, the problem is solved by removing all the leftward arcs[111].

We propose here a new FAS algorithm that tries to minimize the number of removed edges. Our proposed approach is based on two remarks. First, some of the arcs may occur in multiple cycles. As we want to minimize the number of deleted arcs, removing arcs that occur in multiple cycles first will minimize the total number of removed ones. Second, graph nodes in the upper levels usually have more fan-out arcs than fan-in (source nodes have zero fan-in arcs) and vice versa, the nodes on the lower levels of the graph have more fan-in arcs than fan-out (sink nodes have zero fan-out arcs). To keep the graph hierarchy order unchanged, we try to remove the arcs that connect from lower level nodes to upper level ones. So, we suggest using the formula F = FanIn/FanOut to order the nodes. So, nodes in upper levels usually will have a smaller value of F than the nodes in lower levels. Hence, we remove the cyclic arcs that are connected from nodes with higher value of F to nodes with lower value of F. Algorithm 3.2 lists the steps to convert a directed graph to a DAG based on the above remarks.

This algorithm consists of two repetitive steps. The first is to call another recursive algorithm (as listed in Algorithm 3.3) to retrieve the list of arcs to be removed from the graph. The second step is to remove these arcs and check if the graph has become acyclic or not. If it is not acyclic yet, it repeats those two steps. Algorithm 3.3 is a recursive algorithm. It starts by splitting the graph into a set of sub graphs, each of which represents a strongly connected component of the input graph. A strongly connected component is a

Algorithme 3.1 RemoveCycles(H, R)

Input: The adjacency matrix H, the set of root concepts $\mathcal{R}$.
Output: The updated adjacency matrix H^*that represents the DAG of the graph H with $\mathcal{R}$ nodes as roots
Initialization: set $H^* = H$, define the empty closed set $\mathcal{C} \leftarrow \{\}$, define the open set $\mathcal{O} \leftarrow \mathcal{R}$

Begin
    For each $a \in \mathcal{O}$, retrieve the row $H^*_{a,:}$
        Remove a **from** $\mathcal{O}$, $\mathcal{O} \leftarrow \mathcal{O} - a$
        Set$\mathcal{C} \leftarrow \mathcal{C} + a$
        For each non zero node b in the row $H^*_{a,:}$
            If $b \in \mathcal{C}$ **Then**
                Set $H^*_{a,b} = 0$
            Else If $b \notin \mathcal{O}$ **Then**
                Set$\mathcal{O} \leftarrow \mathcal{O} + b$
            End If
        End For
    End For
End

Algorithme 3.2 RemoveMinimumFAS(H)

Input: The adjacency matrix H
Output: The updated adjacency matrix H^*that represents the DAG of the graph H
Initialization: set $H^* = H$,

1: **Begin**

2: **Repeat**

3: Retrieve A = RemoveMinimumFASRecursive(H^*)

4: Remove A from H^*

5: **Until** H^* **is DAG**
6: **End**

sub graph of the graph that contains at least one cycle that connects all the nodes in this sub graph. Going from bigger sub graphs to smaller ones, we create an adjacency H_1 matrix to represent that sub graph. In step 6, we select the set of arcs that occur in a maximum number of cycles. This is done by weighting each of the arcs in H_1 based on the number of the newly reformed strongly connected sub graphs if that arc is removed from H_1, then selecting the arcs with the highest weight and store it in $\mathcal{A}_1$. If $\mathcal{A}_1$ contains multiple arcs, which means that there are multiple arcs with the same weight, then we select the arc a with maximum F where F is defined as F=$\frac{FanIn_{source} \times FanOut_{sink}}{FanOut_{source} \times FanIn_{sink}}$. This means that we select the arc that most probably connects a lower level node to a higher level one. Then, we remove that arc, a, from the sub graph H_1. The sub graph H_1 now contains either no cycles or a smaller set of cycles. So, we call the algorithm again but with H_1 as an input instead of H. The returned set of arcs along with the arc a are added together to $\mathcal{A}$. Then, the last steps are repeated for all other sub graphs. Then the algorithm returns the set of arcs that should be removed in $\mathcal{A}$.

After applying the previous steps, we have finished extracting the representative concepts, $\mathcal{C}$, of our ontology. Also, we have finished creating the taxonomy H of the ontology. Next, we manipulate the knowledge repository articles in order to create concept-term mapping, which represents the core part of the ontology.

Algorithme 3.3 RemoveMinimumFASRecursive(H)

Input: The adjacency matrix H
Output: The set of feedback arcs $\mathcal{A}$ to be removed from H
Initialization: set $\mathcal{A} = \{\}$

Begin
    Get the sets of Strongly Connected Components $\mathcal{S}$ of the graph H
    Sort $\mathcal{S}$ from the bigger to smaller sets
    For each set $S_1 \in \mathcal{S}$ **(ignoring the sets of size** $= 1$**)**
        Generate adjacency matrix H_1 from H including nodes only from S_1
        Select the set of arcs $\mathcal{A}_1$ from H_1 that occur in maximum number of cycles
        If $\mathcal{A}_1$ contains multiple arcs **Then**
            Select the arc a with maximum value of F
        Else
            Set $a = \mathcal{A}_1$
        End If
        Remove a from H_1
        Retrieve $\mathcal{A}_2$ =RemoveMinimumFASRecursive(H_1)
        Set $\mathcal{A} = \mathcal{A} \cup a \cup \mathcal{A}_2$
    End For
End

3.2.2 Manipulating Knowledge Repository Articles

The second part in the creation process of the ontology is to extract the background knowledge associated with each topic in the knowledge repository. Then, we assign this knowledge to the corresponding ontology concept. Each topic in the knowledge repository has a collection of articles that represent the ideas and concepts of that topic as we discussed earlier. The extraction of usable knowledge from these articles is done in several steps. First, we gather all the representative articles of each topic. As we do not care about the articles' structure, we merge all the representative articles of each topic together in the form of one representative article. In some cases, these articles are not sufficient to represent the topic. This situation occurs often in the abstract topics. In this case, we can augment the knowledge associated with each topic with the knowledge associated with it through an ancestral hierarchy of topics. For example, we add the knowledge that are associated with "Anatomy", "Botany", "Zoology", etc., to the knowledge associated with the "Biology" topic. This includes all the topics in the hierarchy up to a specific level l. We used this strategy mainly in the ADTI application, which is discussed in detail in chapter 4.

Note that some of the articles may be merged to multiple representative articles, as they may appear under multiple topics in the knowledge repository. For example, an article that covers "Document clustering" may appear under both the "Data mining" and "Machine learning" topics. After generating the representative article for each KR topic, we apply the next three processing steps for each representative article.

Article Cleaning

Most of the knowledge repositories are available from the web. The articles of these KRs are usually stored in a web format such as HTML, XML, and so on. In this step, we remove all kind of formatting found in the representative articles, such as tags, links, character entity references, and so on. We also remove all the punctuation. The output articles would be in a form of text with alphabetic characters. Then, the articles are broken down into a list of consecutive words. After this cleaning step, the input articles are ready to be used in the next step.

Stop-word Removal

One of the standard steps in document preprocessing is stop-word removal. Stop-words are the list of words that occur frequently in all documents and do not add to the meaning of the document. Some examples of such stop-words are prepositions, articles, pronouns,

etc. There are more than one list of stop-words defined for English which vary in size. Although this step is essential for any text mining task, we have only removed a very small set of common stop-words, such as the determiners. As we will see later, we can use our ontology to determine the most frequently occurring words in the concepts of the ontology, to remove them from the ontology.

Stemming

After removing the basic stop-words found in the articles' lists of words, we stem the words in the articles' lists to their canonical forms. Stemming is the process of converting the input text from its original form to basic or root form. For example, terms such as "economy", "economic", "economist", "economies", and "economize" will be stemmed as "econom". Using stemming as a preprocessing step has become a requirement in any text mining application. It implicitly converts the word to a standard semantic form which represents the meaning of that word whatever its original form (singular, plural, verb, noun, etc.). We used the Porter stemming algorithm introduced in [112]. We call the output of the stemming step terms.

3.2.3 Building the Concept-Term Mapping

After performing the article preprocessing step, the articles contain a combination of stemmed terms. We augment all these lists into one list of terms, $\mathcal{T}$ of size l, which represents the whole ontology list of terms. Then we create for each concept defined in $\mathcal{C}$, extracted in the first step, a vector for all terms found in $\mathcal{T}$. Next, we use the representative processed article, defined in the previous step, for each concept to assign the values for its term vector as follows: if a term t_i does not appear in that list, we put a value of zero in its place in the vector, otherwise we put the number of occurrences, term count, of t_i in its place in the vector. Then, we update the weight for each term t_i, based on its importance to a concept c_j, according to the following equation:

$$\text{tf-icf}_{ij} = \frac{\text{tf}_{ij} \times \log\left(\text{icf}_i\right)}{\sqrt{\sum_{\forall k} \left(\text{tf}_{kj} \times \log\left(\text{icf}_k\right)\right)^2}}$$

where tf_{ij} is defined as the number of occurrences of the term t_i in a concept c_j divided by the total number of terms in that concept, and icf_i is defined as the total number of concepts divided by the number of concepts that contain the term t_i. The denominator of

this ratio is used to normalize the fraction in such a way that each concept vector in the term space will have a unit length. This equation is very similar to the tf-idf equation, which is the most common weighting method used in text mining. Using this equation, we can construct the concept-term mapping matrix M as follows:

$$M = \begin{bmatrix} \text{tf-icf}_{1,1} & \dots & \text{tf-icf}_{1,j} & \dots & \text{tf-icf}_{1,l} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ \text{tf-icf}_{i,1} & \dots & \text{tf-icf}_{i,j} & \dots & \text{tf-icf}_{i,l} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ \text{tf-icf}_{n,1} & \dots & \text{tf-icf}_{n,j} & \dots & \text{tf-icf}_{n,l} \end{bmatrix} \tag{3.5}$$

where n is the total number of concepts that we have extracted in the first step and l is the total number of terms that we have found for all the extracted concepts. Note that, if the term x has not occurred in the concept y, then $\text{tf-icf}_{x,y} = 0$. The previous step is more or less similar to the vector space document modeling approach. Hence, if we have $C = (\overrightarrow{c_1}, \overrightarrow{c_2}, ..., \overrightarrow{c_n})$ as the matrix of all concept vectors and $T = \left(\overrightarrow{t_1}, \overrightarrow{t_2}, ..., \overrightarrow{t_l}\right)$ as the matrix of all term vectors, then we can write the concept representation as follows:

$$C = MT \tag{3.6}$$

At this stage, we have finished discussion of the process of building the ontology. All the previous steps are done just once, and offline. Once the ontology is created, we can use it to accomplish different text mining tasks.

3.3 Using Wikipedia to Construct the WHO

In this section, we apply the approach that we proposed in the previous section using Wikipedia as the knowledge repository. We discuss in this section the reasons for selecting Wikipedia as our knowledge source. We also show how we have applied each of the ontology creation steps to Wikipedia to construct what we call the Wikipedia Hierarchical Ontology, WHO. Lastly, we discuss some of the issues related to Wikipedia as a knowledge repository.

Why Wikipedia?

We have chosen Wikipedia as our Knowledge Repository for the following reasons:

- **Wide range of topics:** Wikipedia has about four million articles contributed by hundreds of thousands of volunteers. Despite that it allows open editing for its articles, this approach shows remarkable quality. A special report published in the journal Nature [92] showed that Wikipedia accuracy compares well with that of Encyclopedia Britannica. Also the size of Wikipedia is growing quickly, which makes it either cover new topics or enrich the already represented topics.
- **Hierarchical structure:** Wikipedia contains a hierarchy of categories which we need to build the taxonomy of concepts. Although the categories hierarchy is not well organized and in some cases it has loops, our suggested method discussed in section 3.2.1 can deal with this problem.
- **Multilanguage support**: Wikipedia supports many languages. This feature can be used to apply our approach to Wikipedia with different languages.

Using Wikipedia as a background knowledge source in different text mining applications was proposed before, as noted in section 2.3.4. Nevertheless, we propose a new approach for handling, organizing, and extracting Wikipedia knowledge in a well structured form of knowledge, an ontology. This ontology can be used in different applications as we will present later.

3.3.1 Extracting WHO from Wikipedia

As we discussed earlier in this chapter, the ontology extraction process is the process of defining the three main components of the ontology: the set of concepts $\mathcal{C}$, the taxonomical concept relations H, and the concept-term mapping matrix M. So, in the next section we start by defining the ontology taxonomy.

Defining WHO Concepts and Taxonomy from Wikipedia

There are two different options when defining the ontology concepts from Wikipedia: either use Wikipedia articles to represent the ontology concepts, or use Wikipedia categories to represent them. There are three main issues with using Wikipedia articles as concepts. The first issue is that the number of articles in Wikipedia is huge, about four million articles, which overwhelms the ontology with concepts and increases the complexity of handling it. The second issue is the coverage of Wikipedia articles. Some of them are very rich and can well represent a topic, but Wikipedia contains many irrelevant articles, which cannot be used as concepts - including them in the ontology introduces too much noise, and the

process of finding and filtering out the irrelevant articles is not an easy task. The last issue in this approach is that the articles do not have a firm inter-relationship that can be used to build the ontological taxonomy. Although there is a line of research which makes use of the articles' links in Wikipedia to build the concepts' relationships, these links still cannot be used to build the taxonomy. On the other hand, the hierarchical structure of the Wikipedia categories are more organized, and contains much less noise than the ones found in articles' links. Also the number of categories is much less than the number of articles, and the coverage of each category is wider than the articles. Hence, we decided to select the Wikipedia categories to represent WHO concepts.

Although our approach is language-independent, we have used the English version of Wikipedia[1]. In the Wikipedia version that we have used, the number of the extracted categories is 565,104. The number of inter-category relations, graph arcs, found between these categories is 1,253,107. We have applied the following steps to clean the hierarchy:

Removing the Minimum Feedback Arc Set We have applied the algorithm listed in 3.2. The number of removed arcs is 546 which represents about 0.04% of the total number of arcs. In order to check the effectiveness of our algorithm, we also applied the algorithm presented in [111]. It takes nearly the same time, 51 seconds compared to 53 seconds for our algorithm, while it suggests removing 1475 arcs, representing about 0.12% of the total number of arcs.

Removing Categories not in the Mainstream Hierarchy Most of the Wikipedia categories falls under one mainstream hierarchy. In order to keep the taxonomy of WHO connected, we remove any category that does not fall in the mainstream hierarchy. This is done by finding the biggest weakly-connected sub graph of the categories graph, and ignoring the other categories. This reduces the number of categories to 546,511.

Removing Categories with No Content There are many categories in Wikipedia that have no articles attached to them. We call them, the categories with no contents. There are two reasons for a category to not have contents. First, the category represents a very abstract idea or topic whose description comes under its descendant categories. The other reason is that the category is not referring to a topic, rather it is just used for grouping or organizing other categories. We remove only the second type of categories and keep the other ones. The first type can be utilized by using its descendants as we will discuss in

[1]We have used the Wikipedia dump version 2010-03-12, other more recent versions of English Wikipedia are available at http://dumps.wikimedia.org/enwiki/

chapter 4. In this type of filter, we do not directly remove the category as it may affect the structure of the taxonomy, rather we just mark them for deletion.

Removing Categories by their Name Many categories in Wikipedia cannot be considered as concepts. They are either used by Wikipedia internally for maintenance and administration issues, such as "Wikipedia articles needing copy edit", or for grouping other categories such as "1950 deaths". Including these categories in the ontology adds a considerable amount of noise. We have defined different filters for these types of categories. All of these filters are based on the categories' names. Here are some examples of these filters: filtering categories whose names contain a year number, in the form of "X by Y", those which end with a highly repetitive word such as "films", and so on. Similar to the previous filter, we mark these categories for deletion.

For the previous two filters, the categories that need to be filtered out from the taxonomy were just marked for deletion. The reason for this is that we do not want to break down the taxonomy by removing these categories. Also, removing a category that occurs between two related categories will implicitly remove the connection between these two categories. Therefore, we have applied a pruning algorithm for these categories. The basic idea of this algorithm is to remove the categories which are marked for deletion and are either sources, without parent categories, or sinks, without child categories. Then we repeat the last step until there are no other sources or sinks marked for deletion. Although some of the categories that are marked for deletion will remain in the taxonomy, we will consider them as dummy concepts which are used for linking other concepts, and they will not be used in the concept-term mapping matrix.

Constructing the Concept-Term Mapping Matrix From Wikipedia

After determining WHO concepts, we define the concepts' properties in the form of the concept-term mapping. We gather the articles that fall under each category in Wikipedia that have a corresponding concept in WHO. Then we clean these articles, and extract the concept-term mapping matrix using the approach discussed in sections 3.2.2 and 3.2.3. One of the steps that we would like to revise here is the stop-word removal module.

Revisiting the Stop-word Removal Module One of the steps of the article preprocessing stage of the proposed approach is stop-word removal. We noticed after generating the concept-term mapping matrix that there are some terms which occur in almost all concepts. These terms cannot be used to differentiate between concepts, thus they are useless in WHO. We considered these terms as stop-words, and hence we decided to remove

them as they also introduce noise to the ontology. We defined a specific concept frequency, $\mathrm{c}f_{th\_hi}$, threshold using heuristics from the concept-term matrix. Any term that occurs in a number of concepts greater than $\mathrm{c}f_{th\_hi}$ is removed from WHO. On the other hand, any terms that occur in very few categories are useless too and introduce noise to the ontology. Hence we defined another specific concept frequency, $\mathrm{c}f_{th\_low}$, where any term occurring in a number of concepts below the threshold is removed from WHO. We can also use the same idea to define a domain specific stop-word removal module.

3.3.2 Wikipedia and WHO Scalability Issue

As noted, Wikipedia's strength comes from the collaborative work done by a large number of volunteer contributors. This causes the size and coverage of Wikipedia to increase rapidly. One of the major requirements of the proposed technique is the scalability of WHO. This means that the knowledge contained in WHO could be easily updated to be consistent with Wikipedia knowledge. We are proposing here a technique which would fulfill this requirement.

In order to check the validity of knowledge in WHO with Wikipedia, we need to examine two different issues: the taxonomy and the concept-term mapping of the WHO. To check the taxonomy, we know that each node in the WHO taxonomy (topic) reflects one of the Wikipedia categories. We can check the $(FanIn + FanOut)$ value for each category. If we find that this value for a specific category is dramatically changed, the difference between the stored value and the Wikipedia value is greater than a certain value, we update the sup graph that contains this category. This will allow us to add new topics to the WHO hierarchy if there are any, after filtering noisy ones as described in the previous section. Also, if this new hierarchy structure has been changed dramatically, we can restructure the WHO hierarchy without affecting its content.

We know that the weighted representative term lists for each WHO node are generated using the combination of all articles under each category. We store with each topic the number of articles which contribute to the knowledge content of each topic. To check the knowledge content of WHO, we can check the number of articles under each category in Wikipedia which reflects a topic in the WHO structure. If the number of articles of a category increased by a certain ratio, we can update the list of representative terms of this category. By applying these two techniques for the taxonomy and the concept-term mapping of the WHO, the knowledge in the WHO will be scalable.

3.4 Ontology Evaluation and Applications

We discussed in section 2.3.2 the different approaches that are used for evaluating the ontology. One of these approaches is the application-based evaluation. In this evaluation approach, the ontology is used for a specific application and the results are compared to the results without using the ontology. If the performance of the application is improved, then the ontology is useful and can be used for this application.

We have used this approach to evaluate our ontology, as our main goal of creating the ontology is to use it to improve the performance of different text mining applications. There are several text mining applications in which WHO can be employed. These applications include measuring the semantic relatedness between terms or topics, document tagging, enriching document modeling, and so on. As we reviewed in section 2.3.3, the most common application in the literature that employs background knowledge in text mining is the document modeling enrichment. This is done by representing each document as a vector in the concept space, instead of the common term space. This means that documents are represented as a weighted average of the concepts they represent. This representation can be used to improve the performance of text mining applications, such as document clustering and document classification. Although this application is not the main goal of this ontology creation, we review in the next subsection how to utilize our ontology in such an application.

Our main application of WHO is Automatic Document Topic Identification (ADTI). In this application, we are given some topics of interest from the user, and asked to find the best matching topic for each input document. In contrast to the document modeling enrichment application, both documents and concepts are represented as vectors in term space. The best matching topic for a document is identified by finding the best matching concept. This approach is discussed in detail in Chapter 4.

3.4.1 Using WHO for Document Modeling

Let $\mathcal{D} = \{d_i : i = 1, ..., m\}$ be a set of input documents that need to be processed. Let $\overrightarrow{d_i}$ be the vector representing the document d_i. Using the Vector Space Model (VSM) notation, the document d_i is represented as a vector $\overrightarrow{d_i}$ as follows:

$$\overrightarrow{d_i} = \sum_{\forall k} w_{ik} \overrightarrow{t_k}$$

where $\overrightarrow{t_k}$ is a vector that represents the term t_k, and w_{ik} is the weight of the term $t_k$in

the document d_i. The most commonly used weighting is tf-idf (described earlier in section 2.2.1.1). Thus a document d_i can be represented as a vector of all its terms' weights. So, the document-term relation can be shaped in a form of a matrix A, where A_{ij} is the weight of the term t_j in the document d_i. Hence, we can express the above equations in a matrix form as follows:

$$D = AT \tag{3.7}$$

where $D = \left(\overrightarrow{d_1}, \overrightarrow{d_2}, ..., \overrightarrow{d_m}\right)$, $T = \left(\overrightarrow{t_1}, \overrightarrow{t_2}, ..., \overrightarrow{t_l}\right)$. In our approach, the concepts are represented as vectors in the term space as shown in Equation (3.6). Here we assume that we can represent the terms T as vectors in the concepts space C by using the M^T. Then, we map the documents to a concept space of the ontology. We can express the concept term relation as follows:

$$T = M^T C$$

where $C = (\overrightarrow{c_1}, \overrightarrow{c_2}, ..., \overrightarrow{c_n})$, and M is the mapping matrix previously defined in Equation (3.5). By substituting T with $M^T C$, the document matrix D defined in Equation (3.7) can be rewritten as follows:

$$D \quad = \quad AM^T C$$

The resultant matrix of the multiplication AM^T can be considered the document-concept mapping matrix. We call this matrix $\tilde{A}$. Hence, we can map the input documents to our ontology concept space by calculating $\tilde{A}$ as follows:

$$\tilde{A} = AM^T$$

where $\tilde{A_{ij}}$ is the weight of the term c_j for the document d_i.

Documents Similarity Measure

Usually the similarity between the vectors is calculated based on their dot product. For two documents a and b, it is defined as:

$$\mathbf{s}\,(a, b) \quad = \quad \frac{\overrightarrow{a} \bullet \overrightarrow{b}}{||\overrightarrow{a}||\,||\overrightarrow{b}||}$$

where $\bullet$ refers to the dot product and $\|\overrightarrow{a}\|$refers to the length of the vector $\overrightarrow{a}$. Since D represents a vector of all documents, we can define the similarity between documents in a matrix form, S, as follows:

$$\begin{aligned} S &= DD^T \\ &= ATT^TA^T \\ &= AGA^T \end{aligned} \tag{3.8}$$

G represents the Gram matrix which is the matrix of all possible inner products of T. It also can be interpreted as the correlation between terms and can be represented as follows:

$$G = TT^T = \begin{bmatrix} \overrightarrow{t_1} \bullet \overrightarrow{t_1} & \overrightarrow{t_1} \bullet \overrightarrow{t_2} & \cdots & \overrightarrow{t_1} \bullet \overrightarrow{t_l} \\ \overrightarrow{t_2} \bullet \overrightarrow{t_1} & \overrightarrow{t_2} \bullet \overrightarrow{t_2} & \cdots & \overrightarrow{t_2} \bullet \overrightarrow{t_l} \\ \vdots & \vdots & \ddots & \vdots \\ \overrightarrow{t_l} \bullet \overrightarrow{t_1} & \overrightarrow{t_l} \bullet \overrightarrow{t_2} & \cdots & \overrightarrow{t_l} \bullet \overrightarrow{t_l} \end{bmatrix}$$

In VSM the matrix G can be assumed to be the identity matrix ($G = I$) as the terms are assumed to be orthogonal. In our approach, the Gram matrix G can be calculated in the new concept space as follows:

$$\begin{aligned} G &= TT^T \\ &= M^TCC^TM \end{aligned}$$

By assuming that the concepts' vectors are orthogonal in an n-dimensional Cartesian space of concepts, $\mathbb{R}^n$, the multiplication CC^T is equal to identity matrix I. Hence, the Gram matrix will be:

$$G = M^TM$$

So by this definition of G and Equation (3.8), the similarity between the documents can be calculated as follows:

$$
\begin{aligned}
S &= AGA^T \\
&= AM^T MA^T \\
&= \tilde{A}\tilde{A}^T
\end{aligned}
$$

We have detailed the use of this document modeling to improve the performance of the document clustering task in [113].

3.4.2 Revisiting the Running Example

Let us revisit the example that we proposed in chapter 1. We introduced five different sentences and showed that by using the conventional VSM, we could not find any similarity between these sentences. Also, we can show that the other standard document modeling approaches such as the GVSM and the LSI will fail to distinguish between the input sentences, as the input data are very small and we could not infer any term correlation from them.

On the other hand, we can show that the WHO document modeling could find some similarity based on these small input documents. We have used the proposed approach to calculate the similarity between the key term of each sentence. We have selected the terms "economy", "unemployment", "government", "politics", and "president" as the key terms for sentences 1, 2, 3, 4, and 5 respectively. Table 3.2 shows a correlation matrix between these terms. This matrix is calculated based on the cosine similarity between the corresponding rows from the matrix M^T.

	economy	unemployment	government	politics	president
economy	100.0%	**9.2%**	2.5%	0.5%	0.3%
unemployment	**9.2%**	100.0%	1.2%	0.7%	0.2%
government	2.5%	1.2%	100.0%	3.3%	**3.7%**
politics	0.5%	0.7%	**3.3%**	100.0%	1.7%
president	0.3%	0.2%	**3.7%**	1.7%	100.0%

Table 3.2: Terms correlation matrix

From the table, we can notice that the nearest term to the "economy" term is "unemployment" and vice versa. Also the term "government" is near to the terms "economy", "politics", and "president" while the nearest term is "president", and so on. We can see from the above table that these percentages of similarity between terms are logical and similar to simple human judgment of the similarity between terms.

3.5 Summary

In this chapter, we first discussed the need for using background knowledge to enrich document representation. We showed how to build an ontology, which contains an organized and structured form of knowledge, from a different format of knowledge repository. At the end of this chapter, we investigated the reasons for selecting Wikipedia as a knowledge repository, and discussed the various difficulties and implementation issues when using it for creating the Wikipedia Hierarchical Ontology, WHO. In the next chapter, we show the main application that we proposed for the ontology created in this chapter.

Chapter 4

Automatic Document Topic Identification Using WHO

This chapter introduces the proposed Automatic Document Topic Identification (ADTI) approach using our constructed ontology (WHO). We start by giving a brief introduction to this approach. Then we describe this approach in detail discussing the different modules and aspects that affect its performance. Finally, we compare it with similar text mining tasks.

4.1 Overview

As we have reviewed in chapter 2, the term "topic identification" has been used as equivalent to "topic indexing", which means predicting the best topic or topics to represent the input document set. In most cases, this prediction is based on some background knowledge extracted from a knowledge repository which is different from the source of the input documents. A good example of these approaches is document tagging. The purpose of the document tagging approach is to predict the best matching tags for each input document. Usually this set of tags is very large and extracted from the knowledge base. Usually each document is assigned multiple tags. In this work, we have a different definition for the term "topic identification". Given that we have a set of topics labels, $\mathcal{L} = \{b_1, b_2, \ldots, b_p\}$, that has been marked as being "of interest", and we have a set of input documents, $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$, and we want to assign each input document $d_i$to one of these topics b_j, we refer to this task as Automatic Document Topic Identification (ADTI).

As we can see from the above definition, we use the word "identification" to mean finding the best match between the input document set and the input topic list. In contrast to other approaches, the list of topics is a known entity in our approach, which means that we do not need to predict them. On the other hand, our approach makes sure that the topic assigned to each document is one of the input list of topics. To better understand the difference, let's consider the following example: let's assume that we are interested in the following topics: economics, politics, and sports, and we have a document declaring "Barack Obama is the new President of the United States". Usually other topic indexing approaches will use some form of background knowledge to predict the topic of this document, and they might give the following topics as the best matching topics: "US presidential election", "Presidents of the USA", and so on. These topics usually will be very specific to this document. In contrast, our approach will try to identify the most relevant topic from the list of given topics, and will identify it as "politics".

ADTI can be used in many different real applications, for example, improving retrieval of library documents pertaining to a certain topic. It can also be used to improve the relevancy of search engine results, by categorizing search results according to their general topic and giving users the ability to choose the domain which is more relevant to their needs. It is also needed for an organization like a news publisher or news aggregators, where they want to automatically assign each news article to one of the predefined news main topics. Similarly, it can be applied for digital libraries to assign each new article to one of the predefined list of topics.

ADTI can also be used to improve the output of an automatic speech recognition (ASR) system. In order to produce the recognized text, an ASR system usually needs to be supplied with a language model. The efficiency of the ASR system is very much dependent on the accuracy of the supplied language model. So if we could know the topic of the speech input, we could supply a more relevant language model. Most of the time we do not know this kind of information in advance, therefore we provide a generic language model, which leads to a lower accuracy of the ASR system. To overcome this problem, we can provide the output of the ASR system using this kind of generic language model to the document identification system. The document identification system will provide in return the most relevant topic using the inaccurate output of the ASR. Consequently, we could supply a more relevant language model, and get a more accurate result when this is applied to the ASR task.

The rest of this chapter is organized as follows: Section 4.2 shows how we utilized our proposed ontology, WHO, in ADTI. This includes the different methods that we used to extract the desired topics from WHO concepts, the different approaches using the WHO taxonomy to improve the topics' representation, and the different proposed identification

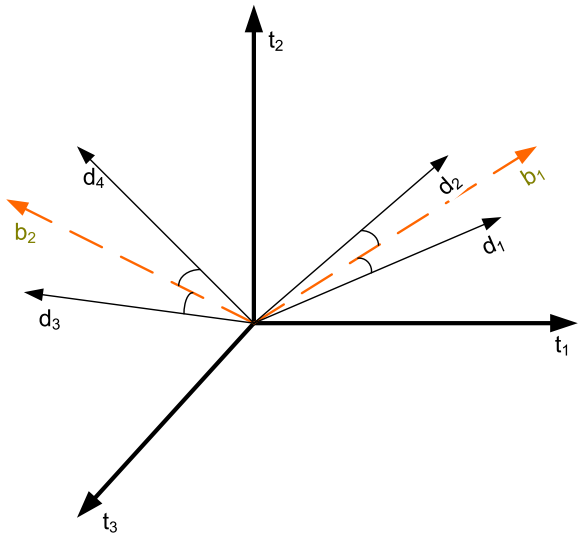

Figure 4.2.1: ADTI basic idea.

approaches to select the best matching topics for the documents. After that, we will do a comparison between ADTI and the other common text mining applications in section 4.3.

4.2 Automatic Document Topic Identification Methodology

As previously noted, the idea of ADTI is to map topics and input documents to the same space, then find the closest topic to each input document. Here the common space between input documents and topics is the term space. Figure 4.2.1 illustrates the concept of ADTI in a graphic presentation. It shows a sample of the representation for four documents' vectors $\left\{\overrightarrow{d_1}, \overrightarrow{d_2}, \overrightarrow{d_3}, \overrightarrow{d_4}\right\}$ and two vectors of topics' labels $\left\{\overrightarrow{b_1}, \overrightarrow{b_2}\right\}$ in a three-dimensional space of terms $\{t_1, t_2, t_3\}$. Using this representation, we can see that the documents' vectors $\overrightarrow{d_1}$ and $\overrightarrow{d_2}$ are closer (based on the cosine similarity) to the topic vector $\overrightarrow{b_1}$ than to topic vector $\overrightarrow{b_2}$. Hence, we can identify the topic of d_1 and d_2 as b_1, and the topic of d_3 and d_4 as b_2.

It can be seen above that the better the representation of the topic in the term space is, the better identification results. So, ADTI is the process of finding the best representation for the input topics, then matching input documents to those topics. We can split this

process into three different steps. The first step is to extract the representative concept vector for each input topic from the constructed ontology. The second step is to use the concept taxonomy in WHO to enrich the topics' representations. The last step is to use extracted topics vectors to identify documents' topics. In the next section, we explore the different ways to extract the representative concept vector that we examined in our approach.

4.2.1 Extracting Representative Concepts for Input Topics

As we stated earlier, one of the inputs of the ADTI application is a list of topics of interest, to which we want to classify input documents. In this module, we try to find the matching list of concepts to these topics. Usually, this process is done by direct matching of topics' names and concepts' names. The problem is that sometimes there is no direct match between some topics and concepts, or the direct matching is not so accurate. For example, sometimes the given topic of interest is "technology", but the best matching concept describing the input document set is "information technology". Or as an example of a topic with no direct matching concept, consider "economy". The best matching concept label for this topic is "economics" not "economy". To resolve this problem, we propose two different approaches to select the best matching concepts for the input topics, manual matching and semantically related automatic matching.

4.2.1.1 Manual Matching

In this approach, we use the data set provider's experience about the input document set to find the matching concepts. Given the list of the ontology concept labels $\mathcal{L}$ that we extracted in section (3.2.1), for each given topic label, we start by searching that list for all available concept labels that contain this topic label. We sort the output list for each topic label based on the orthographic similarity between the topic label and the list of concepts' labels. We pass on these lists of concepts' labels to the data set provider to select the best matching concept(s) for each topic, based on their experience with the input data set. The main drawback of this approach is that it makes the whole technique partially manual, as we still need a human to select the matching concepts.

4.2.1.2 Automatic Matching Using Semantically Related Concepts

In contrary with the manual matching approach, we try to resolve the matching problem automatically. For each input topic, we want to find the set of the semantically similar

concepts from our ontology. We use a lexical database to identify the best matching concepts' labels. As we reviewed in section 2.3.3, WordNet is considered the most famous English lexical database. The proposed approach is as follows. For each given topic label, we extract from WordNet the synsets for that topic. Each of these synsets points to a list of terms that are synonymous to the input topic label; also, each of the output synsets is possibly connected to some other synsets which have a semantic relation to that synset. These semantic relations include, but are not limited to, antonym, hypernym, hyponym, and so on. Here we are interested in the "derivationally related" semantic relation. "Derivationally related" is a relation between a set of terms with similar meanings but in different form, such as "economy" and "economics". We use both the original list of synsets and their derivationally-related synsets to extract a list of possible candidate concepts. We use this list to search for the exact-match concepts from the list of the ontology concept labels, $\mathcal{L}$. The output list of matching concepts is considered the list of representative concepts for that topic. The main drawback of this approach is that the output list of concepts will not be as accurate as in the manual approach, in which we select the concepts manually based on human experience.

After identifying the matching concepts either manually or automatically, we construct the topic-concept map matrix P. Each row of this matrix represents a topic and each column represents a concept. In other words, if the number of the input topics is p and the total number of concepts extracted in WHO is n, then P size will be $p \times n$. Each element of this matrix, $P_{i,j}$, is equal to one when the concept j is considered a representative concept for the topic i, according to the list matching concepts extracted in the previous step, and is zero otherwise. Notice that the matrix P is too sparse and $n \gg p$.

4.2.2 Enhance Topic Representation by Utilizing Ontology Taxonomy

In some cases, this concept-term matrix P does not suffice to represent the topic well. This situation occurs often in abstract topics where the number of relevant articles in Wikipedia is too small, hence the representing concept vector will have a very small list of representing terms. For example, the Wikipedia category "computer science" is only covered by only four articles. Consequently this will affect the identification of the topic. As we introduced in chapter 3, each concept in our extracted ontology, WHO, has a conceptual relationship to other concepts which are represented in the ontology taxonomy, in addition to its representative terms' vectors. We utilize the hierarchical structure of concepts to increase the amount of information that is associated with each topic; this also increases the generality of the topics. We do this by augmenting to the concept-term mapping vector

of each topic of interest, the set of term-mapping vectors associated with each concept under the hierarchy of that topic of interest. For example, if the topic of interest is "Biology", we add term mapping vectors that are associated with the concepts "Anatomy", "Botany", "Zoology", etc., to its associated concept-term mapping vector. This includes not only the directly-connected concepts to this topic, but also all the topics in the hierarchy down to a specific level l.

4.2.2.1 Handling Redundant Sub-concepts

As we have mentioned before, the extracted taxonomy for our hierarchical ontology, WHO, is not a tree structure, rather it is a directed acyclic graph. This means that each concept may have multiple parent concepts. Although it does not contain loops, it may contain multiple paths from one concept to its sub-concepts. We call these sub-concepts the redundant sub-concepts, RSCs. RSCs can also occur if we have multiple concepts representing the main topic and there are some shared sub-concepts of these concepts. Figure 4.2.2 shows four different examples of the redundant sub-concepts for a concept C.

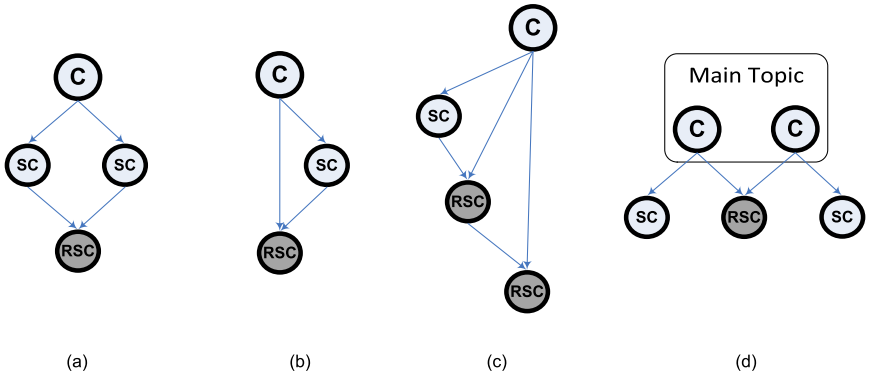

Figure 4.2.2: Redundant Sub-concepts

We have two different options to handle these RSCs. The first option is to consider multiple occurrence of a sub-concept as way of emphasizing this sub-concept. Hence, when we augment the information associated with an RSC to the main concept, we will multiply it by p if there are p paths from the main concept to that RSC. The logic behind this option is that when there are multiple paths between two concepts, this means that these two concepts are highly correlated. Thus, using the adjacency matrix H defined in section (3.2.1), we can define the enrichment matrix E as:

$$E = I + H + H^2 + ... + H^l \tag{4.1}$$

where I is the identity matrix of size $n \times n$. Here we are using the useful fact that the entry $H^k_{i,j}$ of the k^{th} power of the adjacency matrix H counts the number of paths of length exactly k from concept c_i to concept c_j. Hence the summation $H + H^2 + ... + H^l$ represents the total number of paths between two concepts up to length l. Consequently, we can use the concept term mapping matrix defined in Equation (3.5) to define the enriched mapping matrix $M^{(E)}$, where each concept is enriched with the knowledge stored in its sub-tree down to a specific level l as follows:

$$M^{(E)} = EM \tag{4.2}$$

The other option is to consider each RSC only once, so the number of paths between the main concept and an RSC does not count. The logic behind this option is that the paths represent connections between concepts. Thus multiple connections between concepts do not affect the relatedness between these concepts. In this case, we can modify the enriched mapping function $M^{(E)}$ as follows:

$$M^{(E)} = \mathcal{I}(E)\, M$$

where $\mathcal{I}(A)$ is a binarization indicator function that takes a matrix A as input and returns a matrix B of the same size such that

$$B_{ij} = \begin{cases} 1 & A_{ij} > 0 \\ 0 & otherwise \end{cases} \tag{4.3}$$

Hence, if we have $B = \mathcal{I}(E)$, each element of the output matrix B_{ij} is equal to 1 if a path exists from the concept c_i to the concept c_j with maximum length l, or to 0 otherwise.

4.2.2.2 Applying Penalty Function

Although the previous augmentation of sub-concepts' information to the main concept increases the amount of information that is associated with the main topic, it also adds some noise. Noise here means a subset of the information that is related to the sub-topic, but is not related to the main topic. Since the relatedness between the main concept and its sub-concepts decreases as they get farther from it, the quantity of noise increases as we go deeper into the hierarchy of concepts. To resolve this problem, we introduced a penalty function α to penalize the information coming from the sub-concepts. The penalty term should be a function of the level of the sub-topic, so that as you go deeper into the hierarchy the penalty value increases.

To choose the best, we propose three different penalty functions. The first penalty function is the identically one function, $\alpha_1(x) = 1$. This function assigns the same weight to all the hierarchy levels, which makes the penalized version of the enrichment matrix $E^{(\alpha)}$ more or less equivalent to E defined in Equation (4.1). The second penalty function is the multiplicative inverse function, $\alpha_i(x) = {}^1/_{(x+1)}$ where the amount of information added from sub-concepts to the main concept is inversely proportional to their distance from it. The last penalty function is the exponentially decreasing function, $\alpha_e(x) = e^{-x}$. In this function, the amount of information added from sub-concepts to the main concept exponentially decreases with their distance from it.

Going back to the redundant sub-concept issue, the calculation of the penalty function will be straightforward if the RSC occurred at specific level l from the main concept, as in Figure 4.2.2 (a) and (d). On the other hand, if the RSC occurs at multiple levels of distance from the main concept, as in Figure 4.2.2 (b) and (c), we have two options: either consider the RSC as occurring more than once, as in the first option of handling RSCs, or consider the RSC to occur once at the lower/higher level. Hence we can update the enrichment matrix E to include the penalty as follows:

$$E^{(\alpha)} = I + \mathcal{G}\left(\alpha(1) \times \mathcal{I}\left(H^1\right),\ \alpha(2) \times \mathcal{I}\left(H^2\right), \ldots \alpha(l) \times \mathcal{I}\left(H^l\right)\right) \tag{4.4}$$

where $\alpha(x)$ is one of the proposed penalty functions $(\alpha_1(x), \alpha_i(x), \alpha_e(x))$, $\mathcal{I}(A)$ is the binarization indicator function as described in Equation (4.3), and $\mathcal{G}\left(A^{(1)}, A^{(2)}, \ldots A^{(k)}\right)$ is an aggregation function which takes k matrices of the same size and returns a matrix G whose$(i,j)^{\text{th}}$ element is calculated based on the aggregation of the $(i,j)^{\text{th}}$ elements of the k matrices. This aggregation function can be the sum, maximum, minimum, or average of A_{ij}'s elements. Thus, if $\mathcal{G}$ is the sum function, we are considering RSC as occurring more than once. On the other hand, if the $\mathcal{G}$ is the maximum function, we are considering RSC as occurring at the lowest level, and so on.

4.2.2.3 Filtering Concept Sub-tree

As we stated above, the main objective of the ADTI application is to extract a good representation for the topics of interest to better identify the input documents. In order to do so, we have augmented the concepts of the sub-tree of each representing concept to enrich the concept representation. One of the issues that arises as a result of this augmentation is that sometimes the main concepts have a shared set of the sub-concepts such as those which appear in Figure 4.2.3.

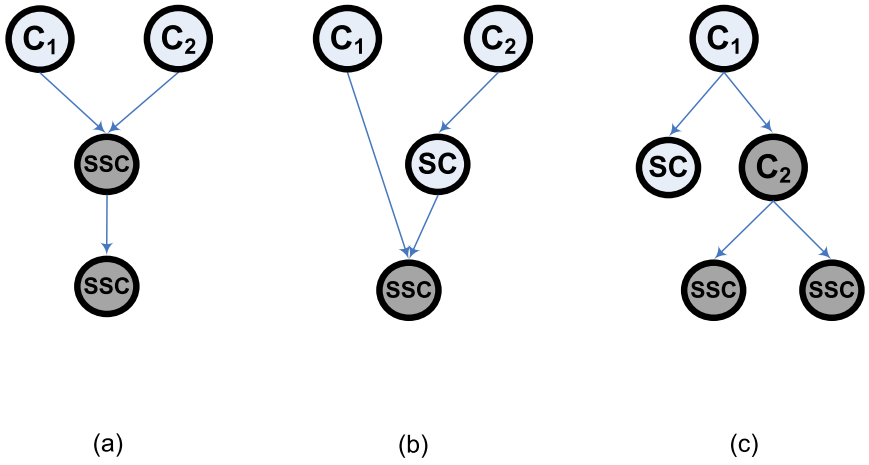

Figure 4.2.3: Shared Sub-concepts

Handling Shared Sub-concepts Shared sub-concepts, or in short SSCs, are the subset of concepts that occur in the sub-tree of any two main concepts. In other words, suppose that C_1 and C_2 are two concepts that represent two of the main topics of interest. If we have a sub-concept SC that occurs under the sub-tree of both C_1 and C_2, then we consider SC as an SSC. Figure 4.2.3 shows three different examples of the shared sub-concepts for concepts C_1 and C_2. In some cases, one of the main concepts occurs underneath another main concept, as in Figure 4.2.3 (c). A good example of this case, which occurred in one of our data sets, are the two main topics of interest "entertainment" and "sports", where in WHO the concept "sports" occurs under the concept "entertainment".

The idea of ADTI is to generate representative concept vectors for the input topics. These concepts' vectors need to be as coherent and discriminative as possible. The existence of SSCs might affect the performance of ADTI, particularly if the number of SSCs is very large, since these SSCs will affect the separation between the representative concepts. One possible way of dealing with this issue is to delete all the shared sub-concepts for each pair of main concepts excluding the main concepts themselves. The main drawback of this method is that it can delete many of the sub-concepts, which will necessarily affect the richness of the representation of the main concepts. For example, in the above example, we will have to delete all the sub-concepts of the "sports" concept, which will lead to a poor representation of the "sports" concept.

We propose a different approach to handle this issue. We augment any SSC to the nearest main concept and then remove it from all other main concepts. For instance, in Figure 4.2.3 (b), as the SSC is closer to main concept C_1 than to the other main concept C_2,

we will augment it to C_1 representative concepts and remove it from C_2 sub-tree (removing the edge from SC node to SSC node in the graph). A good practical example of this case is the concept "political economy" in WHO. This concept falls under both "economics" and "politics" sub-trees but it is closer to the "economics" concept than to the "politics" concept, thus we will augment this concept to the "economics" concept. This is consistent with the meaning of the concept, as "political economy" is more semantically related to "economics" than to "politics".

In the other case in Figure 4.2.3 (c), where the main concept C_2 falls under the other main concept C_1, we will augment all the SSCs to the C_2 concept as they are closer to it, and remove the whole sub-tree starting from C_2 from C_1 sub-tree (removing the edge from C_1 node to C_2 node in the graph). The last possible situation is that where the set of the SSCs are at the same distance from both parents, as shown in Figure 4.2.3 (a). In this case we have two possible handling options: either augment these sub-concepts to both main concepts, or remove them from both main concepts. The main advantage of augmenting them to both main concepts is that these SSCs will enrich both main concepts, and may increase the separation between them and the other main concepts. The main disadvantage is that it will affect the separation between these two concepts themselves.

4.2.3 Identification Approaches

The last step after selecting the list of concepts that will represent the given list of topics is utilizing these extracted concepts to identify the topics of input documents. In this section, we propose two different approaches for identifying documents' topics: the nearest centroid approach, and the k-means-based approach.

4.2.3.1 Nearest Centroid Approach

This approach is very similar to the nearest centroid classification approach discussed in section 2.2.2.1. The idea of the nearest centroid classification is to create a prototype, a centroid in this case, for each class, to then use this prototype to classify the input documents by assigning each input document to the closest prototype. Similarly, we define a prototype for each topic, to use it in identifying input documents' topics. We use the constructed topic-concept map matrix P, defined in section 4.2.1 and the WHO concept-term mapping M, defined in Equation (3.5), to extract the matrix Q that represents the topics of interest prototypes as follows:

$$Q = PM \tag{4.5}$$

The size of the output matrix Q is $p \times l$, where l is the total number of terms in WHO and p is the number of the input topics. Notice that each row in this matrix represents a topic which is a vector of summation of all its representative concepts' vectors. We can also notice that the size of the matrix Q is much smaller than the original matrix M as $p \ll n$, where n is the total number of concepts in WHO. Alternatively, we can use $M^{(E)}$ instead of M, which is the enriched version of M defined in Equation (4.2) with one of the penalty functions defined.

The next step is to construct the document-term matrix A from input documents according to the conventional VSM representation as shown in Equation (2.1). The size of the output matrix A is $m \times l'$, where m is the total number of input documents, and l' is the total number of terms in the input document set. Then we remove all terms in A which are not defined in Q, as we consider them to be out-of-vocabulary terms and vice versa, all terms found in Q and not defined in A are considered to be out-of-interest terms, and are removed from Q. Then we normalize each vector of Q to be a length of 1. Hence, the new mapping matrix will be:

$$\hat{Q} = L^{-1}Q$$

where L is a diagonal matrix whose elements are the lengths of the matrix Q. We can use the following equation to calculate the document-topic similarity S matrix as follows:

$$S = A\hat{Q}^T$$

where each row $S_{i,:}$ in the matrix S represents the similarity between a document i and the list of the topics of interest. Then we can define the identification function $h(x)$ as follows:

$$h(x) = \arg\max_{e \in 1:p} S_{x,e}$$

where $h(x)$ is a function that takes an index of a document x and returns the index of the most similar topic, e.

4.2.3.2 K-means-Based Approach

The k-means clustering algorithm is also considered a prototype-based approach. The main difference between the nearest centroid classification and k-means clustering is that k-means is an unsupervised learning approach. In k-means, the prototypes of the clusters are randomly initialized, and are then updated in an iterative way, as we discussed in section 2.2.2.2. The main drawback of the k-means algorithm is that its performance is

highly affected by the initial centroids' values, which means there is no guarantee that it will converge to the global optimum. We also reviewed some of the literature approaches that suggest appropriate values for the k-means initial centroids. We can observe that most of the reviewed methods try to estimate the best initial centroids for k-means by using some statistical or data mining techniques to either generate them or pick them up from the data. In this section, we utilize background knowledge to identify the best initial clusters' centroids.

We use the topics' prototypes defined in Equation (4.5) as the initial cluster centroids, then we run the k-means algorithm normally as we discussed in section 2.2.2.2. The idea behind this approach is to augment the information that implicitly comes in the data set with the knowledge stored in the ontology. In other words, we start with our background knowledge which is stored in WHO, then we try to adapt this knowledge with the information embedded in data, in the form of the distribution of the data points.

4.3 ADTI vs other Text Mining applications

In this section, we compare the proposed ADTI approach and some of the other text mining approaches, including definitions and goals of each task. Also we compare their learning paradigm, whether supervised or unsupervised learning, and show how to use each of these tasks to identify documents' topics. We start by comparing ADTI with document classification, then with document clustering, and finally with the other topic indexing techniques.

4.3.1 ADTI vs Document Clustering

Document clustering is the process of assigning a set of documents into groups, such that the documents in each group are more semantically similar to each other than to the documents in other groups. This task is usually done by representing the documents in the vector space model (VSM), then applying one of the common data clustering techniques. As we can see from the ADTI definition, ADTI will split the input documents into groups (clusters) based on the given set of topics. In this manner, ADTI is similar to the document clustering task.

On the other hand, ADTI is different from the document clustering task, in that it lets you specify a list of topics that you are interested in beforehand. Then by utilizing background knowledge on these topics, the similarity between these topics and the input

documents is measured, allowing identification of the most relevant topics of these documents. This is different from document clustering, which measures the similarity between the documents themselves. Hence, adding or removing a document to or from the document collection will affect the outcome of the document clustering - whereas in topic identification, the identification of one document will not affect the identification of the others. But these two tasks are similar in that they both depend on similarity measures to partition the input documents.

Lastly, document clustering cannot be used to automatically identify document topics. In order to overcome this problem, we can use one of the cluster labeling approaches to assign labels to each clusters. Typically, the labels are extracted from the vocabulary found in the documents in each cluster. The best representative label for the cluster is that which can summarize the idea (topic) of the cluster and differentiate that cluster from the others. As we can see, the resultant set of labels are extracted from the documents' text. In order to use this approach to identify document topics, we still need to find the mapping between the input set of topics and the extracted set of labels. To better understand the mapping problem, let's consider the following example.

Suppose that we have the topic "Politics" as one of the required topics. After applying clustering and cluster labeling, suppose that the best representative label for one of the clusters is "US Presidential Election". Although that we can see that "US Presidential Election" does represent the "Politics" topic, we still need some approach to find this mapping.

4.3.2 ADTI vs Document Classification

As we reviewed in section 2.2.2.1, document classification is the process of finding the best matching category (class) for each new document on the basis of a training set of documents whose category membership is known. Based on this definition, the document classification goal is very similar to the goal of ADTI, which is finding the best matching category (class) for each new document. ADTI differs from document classification in that it does not require a training set of documents with known membership, using background knowledge to identify document topic instead. In this manner, document classification can be considered a supervised learning approach, while ADTI is considered semi-supervised learning approach, as it only requires the topics' labels to identify their matching documents.

It can be argued that the process of extracting, organizing, and utilizing the background knowledge is considered the training phase of ADTI, so it could be considered as a document classification technique. Although this argument has some superficial plausibility, it fails in

the proving for the following reasons. The first difference between document classification and ADTI is that document classification is not concerned about learning the topics of the documents; instead, it is concerned about finding the match between documents and a set of labels. Sometimes these labels are meaningless, or their representative topics are missing, which means that if we want to identify document topics using document classification we need to first find the match between documents and labels, then we need to find the match between these labels and the input set of topics.

The second difference is that document classification techniques are usually data set-dependent, which means that if we trained a classifier with some data set and then tried to use it to classify another data set, most probably we will get poor results even if both data sets have the same categorization of topics. This is because of the change in the vocabulary used for each data set. This is in contrast with ADTI, which depends on a large coverage of vocabulary used from the background knowledge. The data set dependency also includes the list of topics of the data set. When we train a classifier, we train it with a specific list of topics. If we decide to change this set of topics by adding new topics or removing some of them, we need, in most classification techniques, to retrain the classifier. In ADTI, we need just to select different topics from WHO and use them to identify documents topics.

To sum up, ADTI and document classification tasks have the same general main goal, which is assigning new documents to one of a predefined set of labels. Document classification uses a subset of the input documents with known label membership for training then classifies each new document to one of the labels. Then another technique is needed to find the match between these labels and the input set of topics. In contrast, ADTI uses extracted background knowledge to directly identify the topics of each new document.

4.3.3 ADTI vs Document Topic Indexing Tasks

Document topic indexing is the set of tasks that are concerned with finding relevant topics for a set of input documents. As we reviewed in section 2.2.2.3, they have many different real applications. The most well-known document topic indexing techniques are term assignment, document tagging, and keyphrase extraction. In this section, we compare our proposed approach (ADTI) with these common document topic indexing techniques.

The main difference between these techniques and ADTI is that ADTI requires a set of topics of interest to be given to the algorithm as input in order to identify the documents' topics. So, ADTI can be considered a semi-supervised learning approach. On the other hand, the topic indexing techniques are known to be unsupervised or supervised learning techniques. Also as we showed in section 2.2.2.3, the source of terminology for term assignment is usually extracted from an external thesaurus, and is restricted to those terms

in that thesaurus. In document tagging, the source of terminology is usually unrestricted to any source. As for keyphrase extraction, the source of terminology is restricted to the document vocabulary. In ADTI, the source of terminology is vocabulary restricted to the set of topics given beforehand. As for the number of topics aspect, ADTI usually deals with very few main topics - the number of topics is usually limited to fewer than a hundred whereas for the document topic indexing techniques, the number of the topics is usually much bigger (from hundreds to thousands), as they deal with more specific and detailed topics. Based on the above comparison, we can conclude that ADTI is a totally different task than the predefined topic indexing techniques.

In terms of topic assignment, ADTI finds the best matching topic, thus it is considered a single topic assignment approach, while the document topic indexing techniques usually assign multiple topics to each document. Hence, the performance measures used for ADTI and for other document topic indexing approaches are different. Consequently, we would not be able to compare ADTI performance with the other document topic indexing techniques.

4.4 Summary

We have introduced in this chapter a new approach for document topic identification. the proposed Automatic Document Topic Identification (ADTI) approach using our constructed ontology (WHO). We started by giving a brief introduction to this approach then described it in detail discussing the different modules and the aspects that affects its performance. Next, we compared it with similar text mining tasks.

Chapter 5

Experiments and Results

In this chapter, we test the performance of our approach for topic identification. In order to do that, we have conducted a comparative study between our approach and the competing text mining approaches which were previously discussed. In Section 4.3, we showed that the task of document clustering is similar to the task of the ADTI, as each tries to group the input documents into groups of semantically related topics. Also, we showed that the task of document classification is similar to the task of the ADTI, as each tries to find the category membership of input documents. Hence, we have compared the performance of our proposed approach for ADTI to that of conventional document clustering and document classification. Section 5.2 shows the comparative experiments, results, and discussions of our proposed approach and a set of document clustering techniques. Next, section 5.3 does the same with a set of document classification techniques. We conducted these experiments using a number of real-world benchmark document sets. We overview these data sets and their properties in the following section.

5.1 Experimental Setup

5.1.1 Data sets

The selection of data sets that can fit ADTI requirements is more complicated than that for document clustering and document classification. We need data sets that are split into groups or classes, each of which explicitly represents a specific topic. Many of the benchmark data sets that are available for text mining do not meet this requirement. Some data sets are not split according to topics, but rather according to some other feature such

as the author name, the year of publication, and so on. This type of data sets do not fit ADTI. Other data sets are split according to their topics, but the topic label of each class is unknown; an example of these data sets is the fbis data set, which is a part of the TREC collection.

We have selected eight different benchmark data sets that fit ADTI requirements. Most of these data sets have been previously used by Zhao and Karypis [114] to evaluate the performance of different document clustering algorithms. Their properties are summarized in table 5.1. The reviews, sports, hitech, and mm data sets are derived from the San Jose Mercury newspaper articles that are distributed as part of the TREC collection[1]. The k1b and wap[2] data sets are from the WebACE project [115]. Each document corresponds to a web page listed in the subject hierarchy of Yahoo![3]. These data sets were preprocessed and distributed with the CLUTO Toolkit [116]. The preprocessing steps which have been applied to these data sets are stop-word removal and stemming. The bbc and bbc-sports data sets are published from the Machine Learning Group (MLG) at UCD School of Computer Science and Informatics, Dublin, Ireland [117]. The bbc data set consists of documents from the BBC news website corresponding to stories in five topical areas from 2004-2005. The bbc-sports consists of documents from the BBC Sport website corresponding to sports news articles in five topical areas from 2004-2005. We have preprocessed these last two data sets in the same manner as the rest. In all data sets, terms that appear in only one document or do not appear in our ontology, WHO, are removed. Hence, we have n' in table 5.1 to represent the total number of terms after removing these terms. Then, the term weights inside documents are normalized according to the TF-iDF weighting. Lastly, all documents are normalized to represent unit vectors in different directions in term space.

5.1.2 Development Environment and Tools

ADTI development consists of two parts. The first part is the creation of the ontology. We used Microsoft Visual C# as the programming environment, and had the help of the parsing tools that are available at Wikimedia to convert Wikipedia to our ontology, WHO. We stored WHO in an SQL server database. Then we extracted the ontology knowledge to MATLAB format in order to process it. We used the built-in functions in MATLAB version 7.13 (R2011b) for hierarchical clustering. Simulations were carried out on an IBM compatible PC with Intel (R) Core(TM) i5 CPU 3.20GHz and 16.0GB RAM.

[1] http://trec.nist.gov

[2] The original # of documents of the wap data set was 1560 and the # of classes was 20, but we removed the documents, which their classes do not represent topics.

[3] http://www.yahoo.com.

Data set	Source	m	n	$n^{'}$	Topics	k
k1b	WebACE	2340	21839	19021	business, entertainment, health, politics, sports, tech	6
wap	WebACE	1311	8460	7293	people, television, health, media, art, film, business, culture, music, politics, sports, entertainment, industry, multimedia	14
reviews	San Jose Mercury (TREC)	4069	36746	32921	food, movie, music, radio, restaurant	5
sports	San Jose Mercury (TREC)	8580	27673	24115	baseball, basketball, bicycle, boxing, football, golf, hockey	7
hitech	San Jose Mercury (TREC)	2301	22498	20268	computer, electronics, health, medical, research, technology	6
mm	San Jose Mercury (TREC)	2521	29973	27262	movie, music	2
bbc	BBC News	2225	9636	7992	business, entertainment, politics, sport, tech	5
bbc-sports	BBC Sport	737	4613	3804	athletics, cricket, football, rugby, tennis	5

Table 5.1: Summary of data sets used to evaluate the performance of ADTI, m is the number of documents, n is the total number of terms in all documents, $n^{'}$is number of used terms where the other terms that are not present in WHO are ignored, and k is the number of topics.

5.1.3 ADTI Parameters Tuning Experiments

As we discussed in section 4.2, ADTI has some parameters that may affect its performance. In this section we study them to determine their appropriate values so that we can use them in the next comparative experiments. We can sum them up to the following five parameters:

Topic-concept Matching

As we described in section 4.2.1, there are two methods of finding the matching concept/concepts for each given topic: Manual and Automatic matching. We have tested both approaches on sub-sets of the data sets. We have found that Manual topic matching has better results in most cases. We also noticed that as we use more levels of sub concepts to represent topics, the performance of the two approaches becomes more similar. Based on these observations, we will use the manual matching in the comparative study.

Number of Levels

As we discussed before, we use the taxonomical relations between concepts to enrich topics' representative concepts. This parameter corresponds to the number of levels of sub-concepts that we will use to enrich the representative concepts. In order to determine the best value of this parameter, we conducted a small experiment of sub-set of the data sets. We have tried values from 0 to 5 for this parameter, where zero means we included no sub-concepts, and thus no enrichment. We have used the F-measure performance measure for this experiment. Figure 5.1.1 shows the average improvements of F-measure as the number of levels increases, over ADTI performance with zero levels (no enrichment). As can be seen, the performance of the ADTI approach increases with the increase of the number of levels for enrichment. We can also notice that the gain in performance diminishes as the number of levels increases. Figure 5.1.2 shows the average running time of ADTI with different numbers of levels. As we can see the efficiency of ADTI decreases exponentially as the number of levels increases. So there is a tradeoff between the performance of ADTI and its efficiency when increasing the number of levels. Hence, we select the value three for the number of levels as the best value for this tradeoff.

Penalty and Aggregation Functions

In section 4.2.2.2, we have proposed the final enrichment matrix as represented in Equation (4.4). In this equation, we have defined two functions: the penalty function α and the

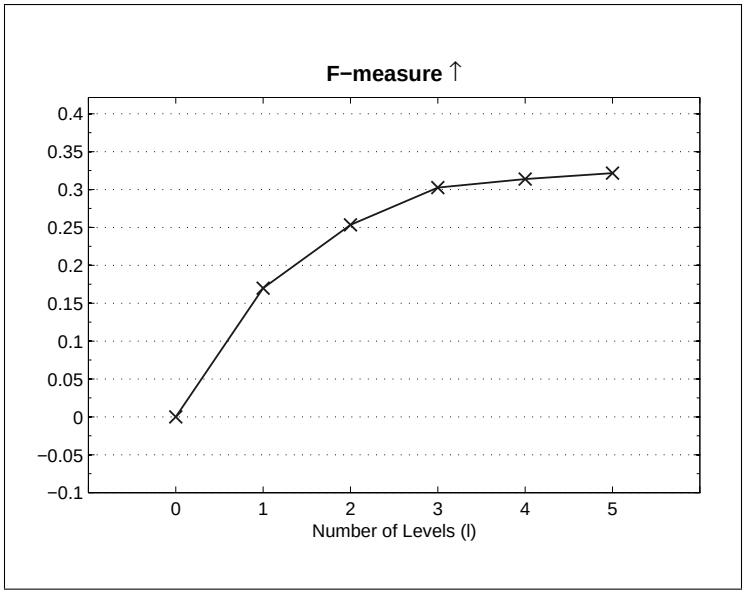

Figure 5.1.1: The average improvements of F-measure with increasing the number of levels over the non-enrichment case

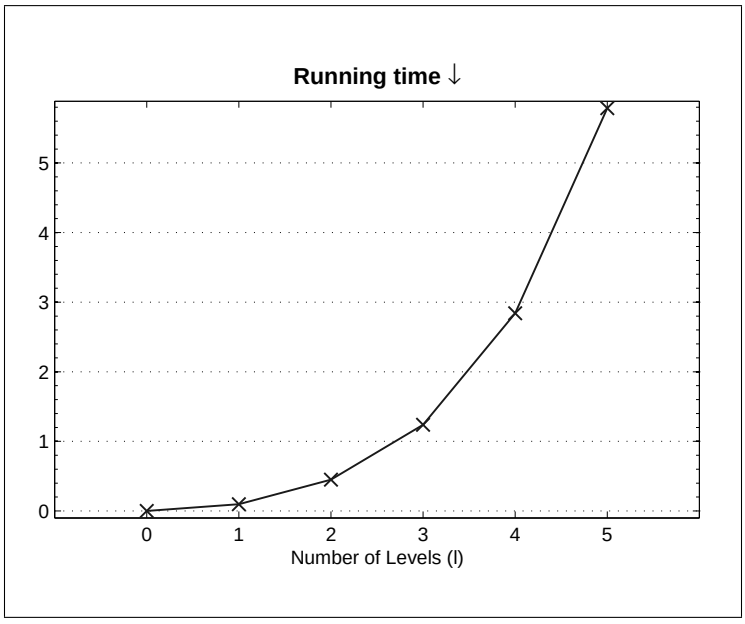

Figure 5.1.2: The average of the running time of ADTI with increasing number of levels

aggregation function $\mathcal{G}$. We have proposed three different penalty functions which are α_1, α_i, and α_e. We also proposed four different aggregation functions which are summation, maximum, minimum, or average. We have studied the effect of these two parameters together along with the number of levels as they are correlated to each other. We have found that the effect of the penalty and aggregation functions on the performance of ADTI is very low when we use a small number of levels, typically from one to three. When we increase the number of levels we found that the penalty function α_e has the best performance, along with the summation aggregation function. Hence, we used these two functions in all experiments.

Handling Redundant Sub-concepts

We can handle the redundant sub-concepts by either removing them from the concepts' sub-trees or by keeping them. Removal is by applying the binarization function as described in Equation (4.4). After testing both options, we found that this parameter also has a low effect on the performance of ADTI when we use a small number of levels. We found also that with a large number of levels, removing the redundant sub-concepts increases the overall performance.

Handling Shared Sub-concepts

As we discussed earlier, the shared sub-concepts are handled by measuring the taxonomical distance between the shared sub-concept and the main concepts, and assigning the shared sub-concept to the nearest main concept. A problem occurs when both main concepts have the same distance to the shared sub-concept. We have one of two options: either keep or remove this shared sub-concept. After testing these options on sub-sets of the data sets, we found that in most cases this parameter has no effect on the performance as the number of the shared sub-concepts is very low, but keeping the shared sub-concepts has overall better performance.

5.1.4 Result Summarization

In order to compare the overall performance of the proposed approach against other approaches and to see how significant the difference is, we need a method to summarize the results over all different data sets. The direct way of averaging the performance measures over data sets will obtain biased results, as the range of measures may vary from one data set to another. We used the summarization technique proposed by Zhao and Karypis

[118]. For each performance measure, p, in a specific data set, if the value of p is directly proportional to the performance, then p is updated by dividing it by the best performing method value for that measure. Otherwise, p is updated by dividing the best performing method value for that measure by p. Suppose that we have the vector p that represents the performance measure p of different methods for a specific data set. So, in the case that the value p is directly proportional to the performance, such as F-measure, the vector p is updated as follows:

$$\tilde{p} = \frac{p}{\max(p)}$$

Otherwise, if p is inversely proportional with the performance, such as entropy, the vector p is updated as follows:

$$\tilde{p} = \frac{\min(p)}{p}$$

Hence, the resultant vector $\tilde{p}$ represents the relative performance for the measure p between methods in a specific data set. These ratios are less sensitive to the actual values of p for a particular data set. The values of $\tilde{p}$ ranges from 0 to 1. The closer the value to 0, the worse the performing method; the best performing method has a value of 1. Now, we can average the values of each method over all data sets to get the overall relative performance $\tilde{P}$ for each method. We can use $\tilde{P}$ to evaluate the statistical significance of the relative performance of each competing method.

5.2 Comparing ADTI to Document Clustering

To test the performance of the proposed approach for ADTI, we have conducted a comparative study with competing text mining tasks. In this experiment, we compare ADTI with four different both standard and state-of-the-art clustering techniques. After applying ADTI and the document clustering, the output labels are compared against ground-truth labels to evaluate each method. In the case of document clustering, we need to find the mapping between the clusters' labels and the provided ground-truth labels. As we mentioned in section 2.2.3, we have used the Hungarian algorithm to find this matching [52]. In the case of ADTI, we do not need to find the matching as ADTI not only partitions the data, but also provides the topics of these partitions. Hence, we can match these labels with the provided ground-truth labels. Both approaches were applied to the eight data sets mentioned earlier.

Techniques

We compared the performance of four different document clustering approaches with the proposed two different variants of ADTI approaches:

- **K-means clustering (K-MEANS):** The spherical k-means version is used since the documents are represented as vectors, and the distance measure used is the cosine similarity. We used the MATLAB implementation of the Lloyd's algorithm[119]. As is well-known, k-means clustering output depends on the initial step. Hence, the cluster assignments are changed for the different runs. So, we applied the k-means clustering 10 times. We report here the mean and standard deviation of these runs.

- **Hierarchical clustering:** Two different linkage methods are used in hierarchical clustering, average (***HIC-AVG***) and complete (***HIC-CMP***) linkage. We also used the MATLAB implementation for the hierarchical clustering. As the hierarchical clustering does not depend on any initial conditions, there is no need to apply it multiple times.

- **Spectral clustering (*SC*):** We have used the cosine similarity as the measure of similarity documents. Regarding the Laplacian matrix normalization, we have used the Ng et al [38] proposed approach: $L = I - D^{-1/2}SD^{-1/2}$. We have used the same k-means implementation for clustering. As this method depends on k-means approach, the cluster assignments are changed for the different runs. We applied this algorithm 10 times as for k-means, and we report the mean and standard deviation of these runs.

- **NMF clustering (*NMF*) :** We have used the Xu et al [41, 42] approach for NMF clustering. As factorization of matrices is generally non-unique, NMF is considered a non-deterministic approach. Hence, we applied the NMF clustering algorithm 10 times and we report the mean and standard deviation of these runs.

- **ADTI:** The two different variations of the proposed topic identification approaches are used in this comparison: ADTI with nearest centroid identification approach (***ADTI-CENT***$_l$) and ADTI with k-means-based identification approach (***ADTI-KMEAN***$_l$), where l is the number of levels used for the method. We reported here the results of these techniques using three levels of enrichment. For the other ADTI parameters, we selected the most appropriate values for these parameters according to section 5.1.3. We used the manual topic-concept mapping, the exponential penalty function α_e and the **sum** aggregation function. Neither the redundant nor the shared

sub-concepts are handled. Both ***ADTI-CENT***$_l$ and ***ADTI-KMEAN***$_l$ techniques have deterministic outputs, therefore multiple runs are not needed.

5.2.1 Performance Measures

In order to judge the performance of our topic identification method against the document clustering approaches, we have to use performance measures that are applicable for both methods. As discussed in section 2.2.3 some of the external performance measures that are commonly used for document clustering, these measures can be used also for ADTI. It is meaningless to use the clustering internal performance measures with ADTI, as the partitioning between the documents is not based on the pairwise similarity between documents as in document clustering, and the aim of ADTI is to identify the topics of documents, not to minimize the separation between classes nor to increase the compactness of each class. In this experiment, we used F-measure (**Fm**), precision (**P**), recall (**R**), purity (**Pu**), **NMI**, and entropy (**E**) to compare the performance. We also measured running time, **T**, of each technique to compare the efficiency of these approaches.

5.2.2 Results and Discussions

Figures 5.2.1, 5.2.2, 5.2.3, 5.2.4, 5.2.5, and 5.2.6 show the output performance measures using five different document clustering approaches and our proposed approaches for eight data sets. They also show the standard deviation of each measure as error-bars with each average value for the measure. Note that the hierarchical approaches and the proposed approaches have no error-bars as they have deterministic outputs.

Regarding F-measure results, we can see from Figure 5.2.1 that ADTI with its both approaches outperforms both hierarchical clustering techniques for all data sets. We can also see that ADTI-KMEAN$_3$ outperforms all other partitional clustering methods for five data sets and has a competitive performance in hitech, bbc, and bbc-sports data sets with both spectral clustering and k-means. Similarly for ADTI-CENT$_3$, we can see that it outperforms the partitional clustering methods in five data sets. In the hitech data set, ADTI-CENT$_3$ has a competitive performance with partitional clustering methods. For bbc and bbc-sports data sets, we can see that ADTI-CENT$_3$ has a competitive performance with NMF, and poorer performance than both spectral clustering and k-means.

Regarding recall measure results, Figure 5.2.3 shows more or less the same performance as F-measure. Figure 5.2.2 shows precision performance for different approaches. It can be seen that ADTI-KMEAN$_3$ outperforms all clustering methods in five data sets,

has a competitive performance with partitional clustering approaches in bbc and bbc-sports, and poorer performance than both hierarchical clustering with average linkage and NMF in hitech data set. ADTI-CENT$_3$ has more or less similar precision performance as ADTI-KMEAN$_3$, except that ADTI-CENT$_3$ performed worse than spectral clustering and k-means in bbc and bbc-sports data sets.

Regarding purity measure, Figure 5.2.4 shows that the proposed ADTI approaches have broadly similar performance to partitional clustering methods, except with the mm data set, where ADTI approaches outperform all clustering approaches. We can also see that the proposed ADTI approaches outperform both hierarchical clustering approaches in all data sets. Figure 5.2.5 shows the output performance comparison with NMI measure. ADTI-KMEAN$_3$ outperforms all different clustering techniques in all data sets. As for ADTI-CENT$_3$, it also outperforms the hierarchical clustering methods too in all data sets and outperforms all different partitional clustering techniques in five data sets, while it has a very competitive performance with the partitional clustering techniques in second data set, but has poorer performance on the last two data sets. Entropy output performance is shown in Figure 5.2.6. The performance of the proposed approaches for this performance measure is more or less similar to the NMI performance measure, except for the first two data sets where ADTI-CENT$_3$ performs similarly to the partitional clustering.

Figure 5.2.7 shows the running time comparison of these approaches. We can see that NMF is the slowest approach. As for ADTI approaches, they have nearly the same running time to most clustering approaches in almost all data sets.

Table 5.2 shows the overall relative performance measures and running time for the different document clustering methods and ADTI approaches. The best output performance is shown in bold and the second best is underlined. This table summarizes the previous results using the approach discussed in section 5.1.4. We can see that our ADTI proposed methods outperforms all the different document clustering techniques in all overall relative performance measures. In terms of running time, ADTI-CENT$_3$ is better than all clustering except the spectral clustering, while ADTI-KMEAN$_3$ has average running time similar to that of K-MEANS and hierarchical clustering.

In order to validate these observations, we evaluated the statistical significance based on the paired sample one-tailed t-test. As we only care about how our proposed methods perform against document clustering techniques, we conducted a pairwise comparison between both ADTI-CENT$_3$ and ADTI-KMEAN$_3$ and each document clustering technique. In this test, each overall relative performance measure for both competing methods, M$_1$ and M$_2$, are tested against each other. If the average performance of M$_1$ is higher than M$_2$, we evaluate the right tailed t-test, otherwise we evaluate the left tailed t-test. If the null-hypothesis is accepted, both methods are considered statistically equivalent. In all

	$\tilde{Fm}$	$\tilde{P}$	$\tilde{R}$	$\tilde{Pu}$	$\tilde{NMI}$	$\tilde{E}$	$\tilde{T}$
HIC-AVG	0.40±0.21	0.71±0.16	0.48±0.18	0.61±0.20	0.43±0.37	0.35±0.20	0.54±0.29
HIC-CMP	0.35±0.13	0.52±0.10	0.40±0.13	0.55±0.13	0.24±0.19	0.30±0.20	0.54±0.29
***K*-MEANS**	0.79±0.17	0.81±0.13	0.79±0.16	0.91±0.10	0.78±0.24	0.70±0.22	0.56±0.23
SC	0.75±0.23	0.82±0.14	0.76±0.21	0.88±0.13	0.73±0.30	0.64±0.25	**0.83±0.34**
NMF	0.72±0.19	0.81±0.14	0.72±0.19	0.84±0.14	0.70±0.27	0.55±0.23	0.09±0.09
ADTI-CENT$_3$	0.95±0.04	0.93±0.08	0.95±0.05	0.94±0.05	0.88±0.08	0.69±0.28	0.64±0.32
ADTI-KMEAN$_3$	**0.97±0.06**	**0.94±0.11**	**0.98±0.04**	**0.99±0.03**	**0.99±0.03**	**0.99±0.04**	0.53±0.24

Table 5.2: The overall relative performance measures for the different document clustering methods and ADTI approaches

t-tests, we use a confidence interval of 95%. The results of this comparison are shown on Table 5.3. $\gg$, $\ll$, or $\equiv$ indicate that M_1 is significantly superior, inferior or equivalent to M_2, respectively.

From Table 5.3, we can observe that ADTI-KMEAN$_3$ performance is significantly superior to all other document clustering approaches. We can also observe that ADTI-CENT$_3$ performance is significantly superior to hierarchical clustering approaches, and it is either significantly superior or equivalent to partitional clustering, depending on the measure used. We can also observe that NMF is the most inefficient approach, while ADTI-CENT$_3$ and ADTI-KMEAN$_3$ have more or less equivalent efficiency to the document clustering techniques.

M_1	M_2	$\tilde{Fm}$	$\tilde{P}$	$\tilde{R}$	$\tilde{Pu}$	$\tilde{NMI}$	$\tilde{E}$	$\tilde{T}$
ADTI-CENT$_3$	HIC-AVG	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\equiv$
	HIC-CMP	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\equiv$
	K-MEANS	$\gg$	$\gg$	$\gg$	$\equiv$	$\equiv$	$\equiv$	$\equiv$
	SC	$\gg$	$\gg$	$\gg$	$\equiv$	$\equiv$	$\equiv$	$\equiv$
	NMF	$\gg$	$\gg$	$\gg$	$\gg$	$\equiv$	$\equiv$	$\gg$
ADTI-KMEAN$_3$	HIC-AVG	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\equiv$
	HIC-CMP	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\equiv$
	K-MEANS	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\equiv$
	SC	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\ll$
	NMF	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$

Table 5.3: Comparison between ADTI approaches and different document clustering methods based on statistical significance (using t-test)

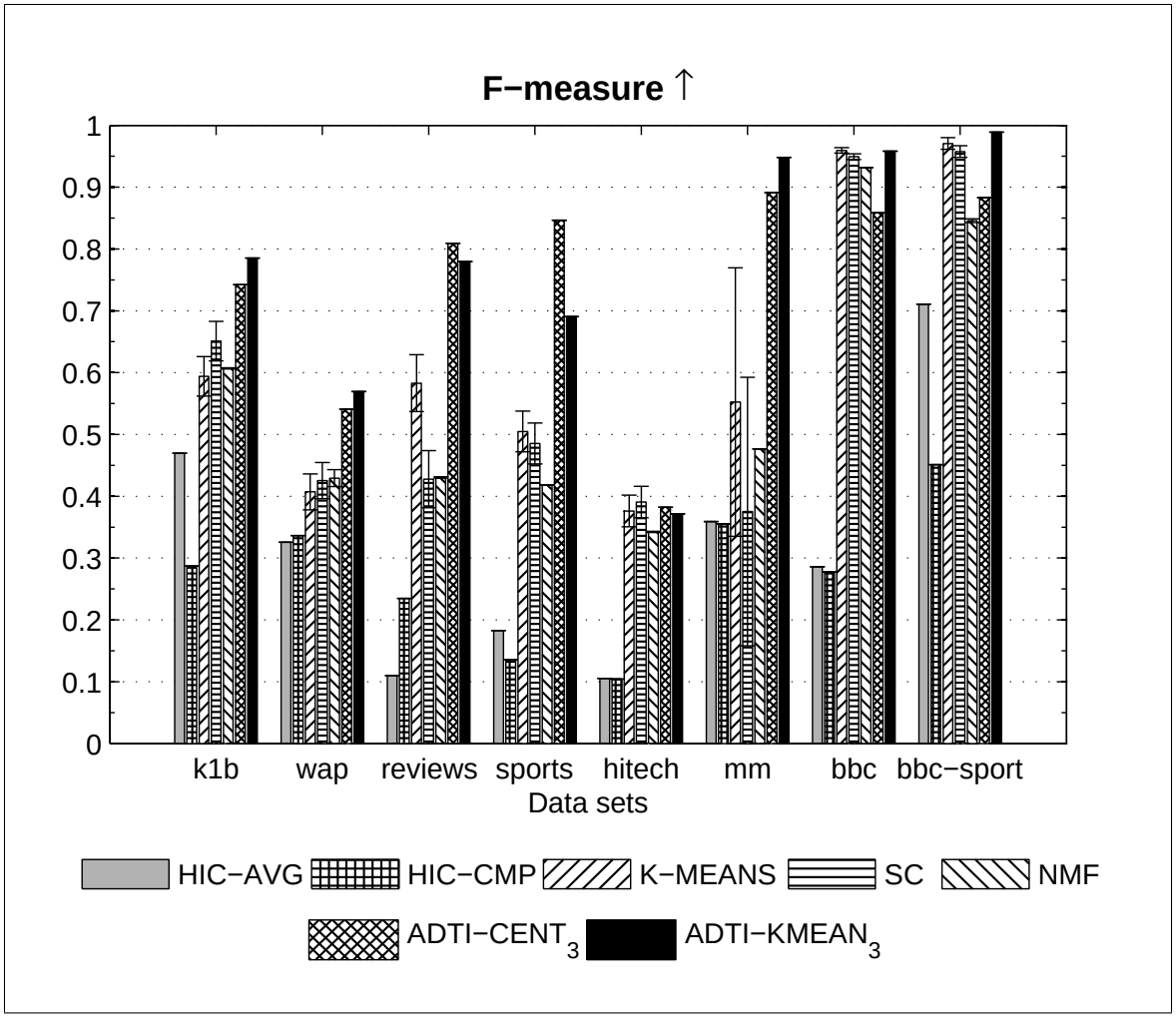

Figure 5.2.1: The output F-measure of the different document clustering methods and ADTI approaches for 8 data sets.

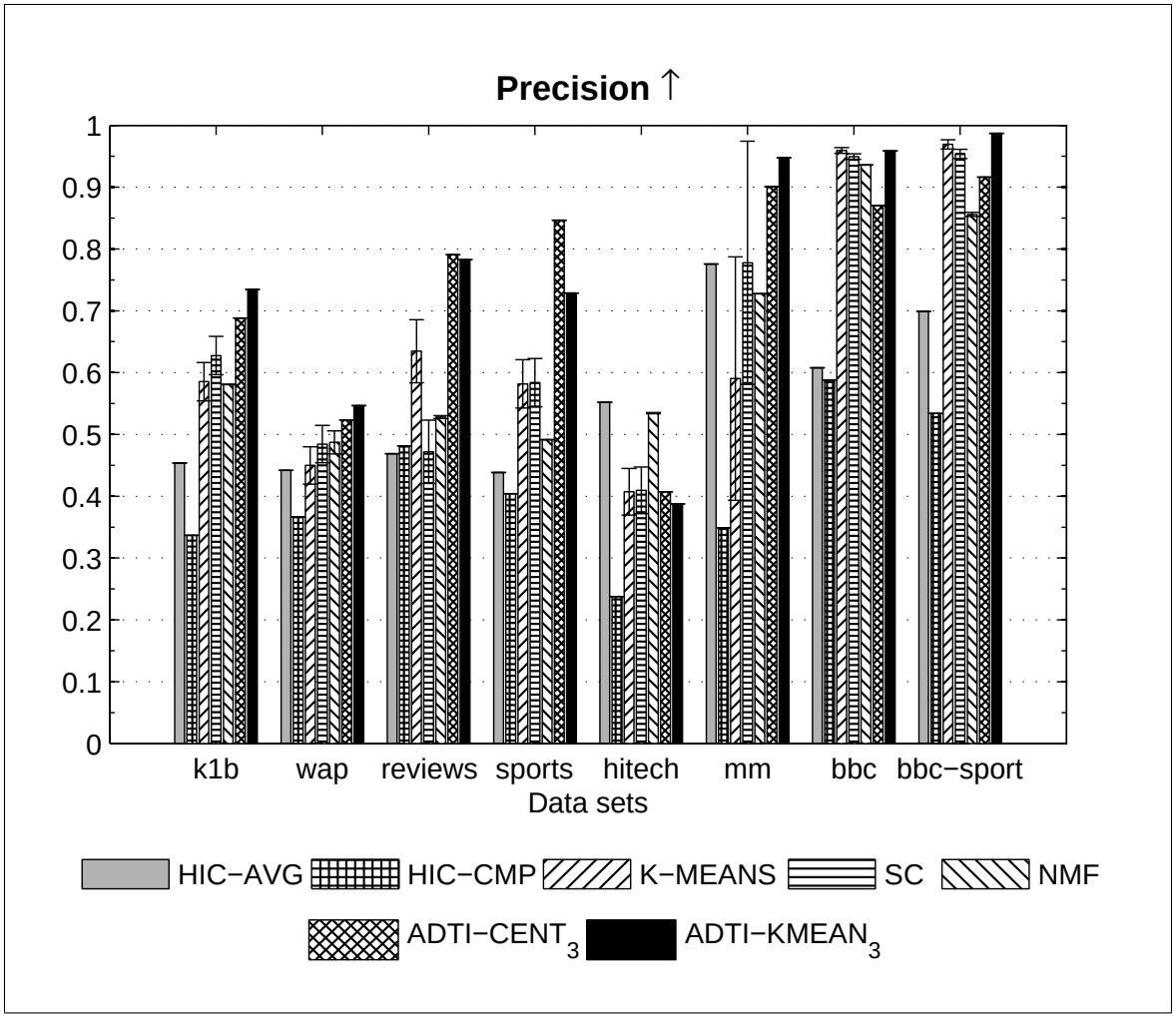

Figure 5.2.2: The output precision of the different document clustering methods and ADTI approaches for 8 data sets.

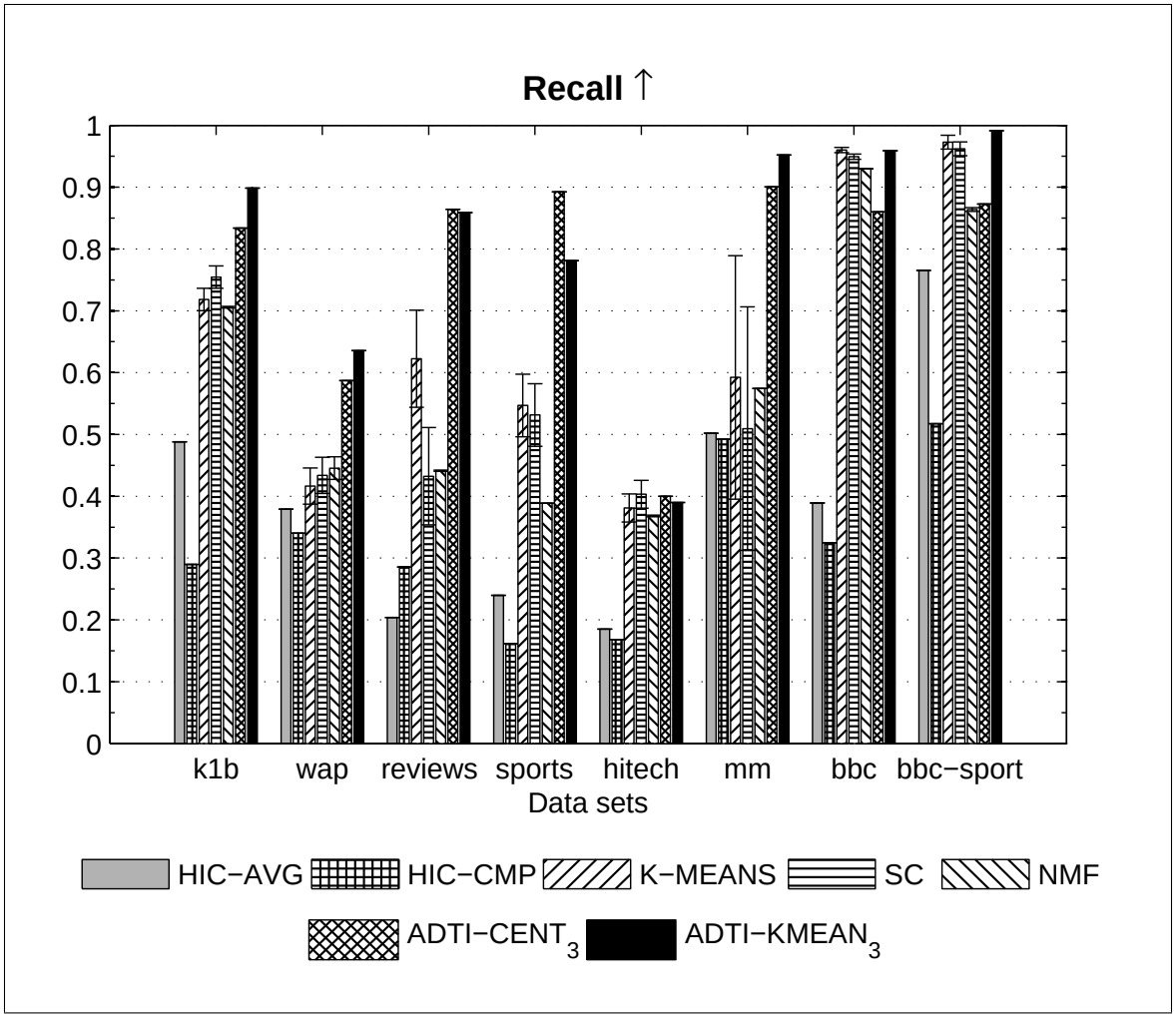

Figure 5.2.3: The output recall of the different document clustering methods and ADTI approaches for 8 data sets.

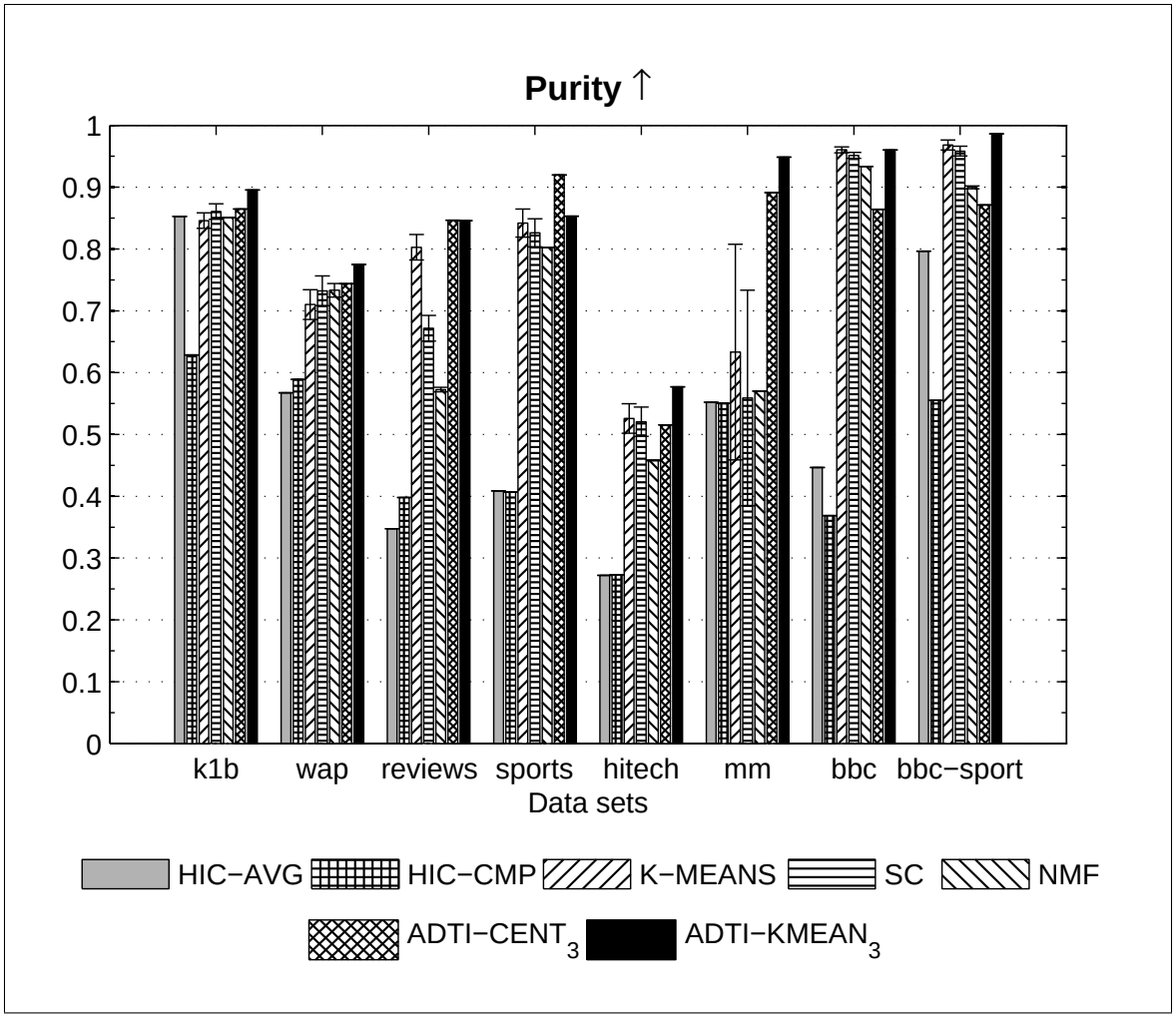

Figure 5.2.4: The output purity of the different document clustering methods and ADTI approaches for 8 data sets.

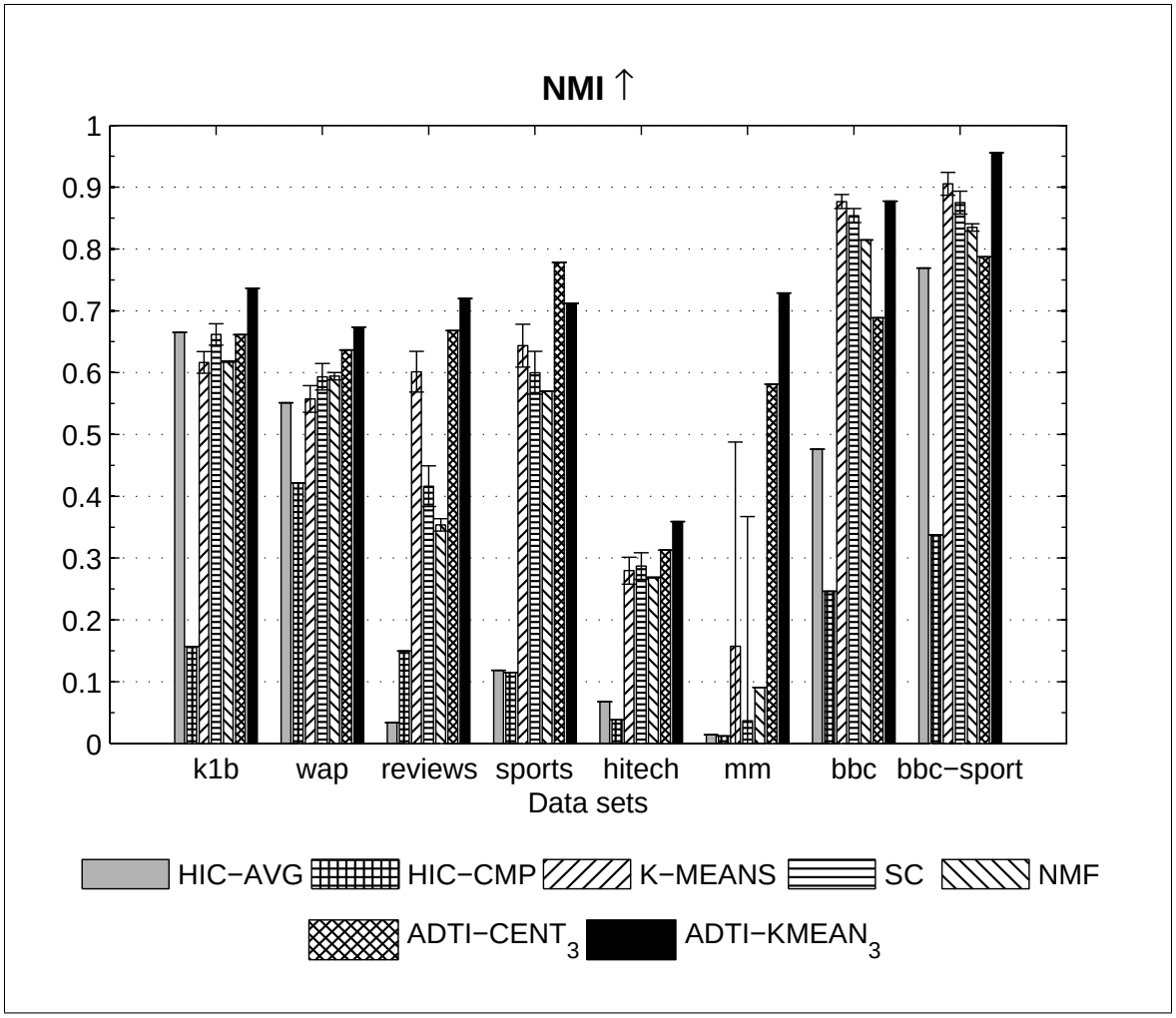

Figure 5.2.5: The output NMI of the different document clustering methods and ADTI approaches for 8 data sets.

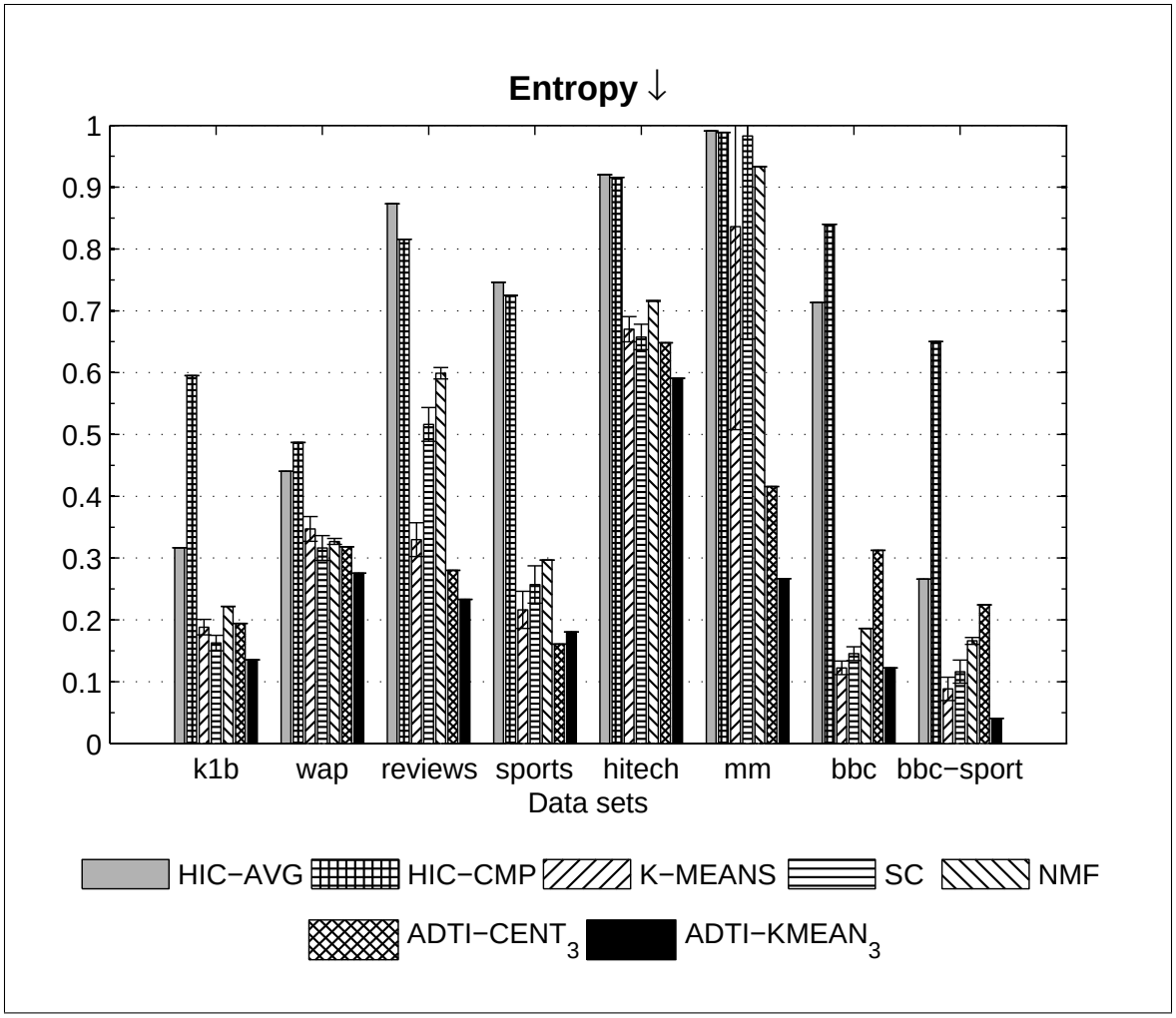

Figure 5.2.6: The output entropy of the different document clustering methods and ADTI approaches for 8 data sets.

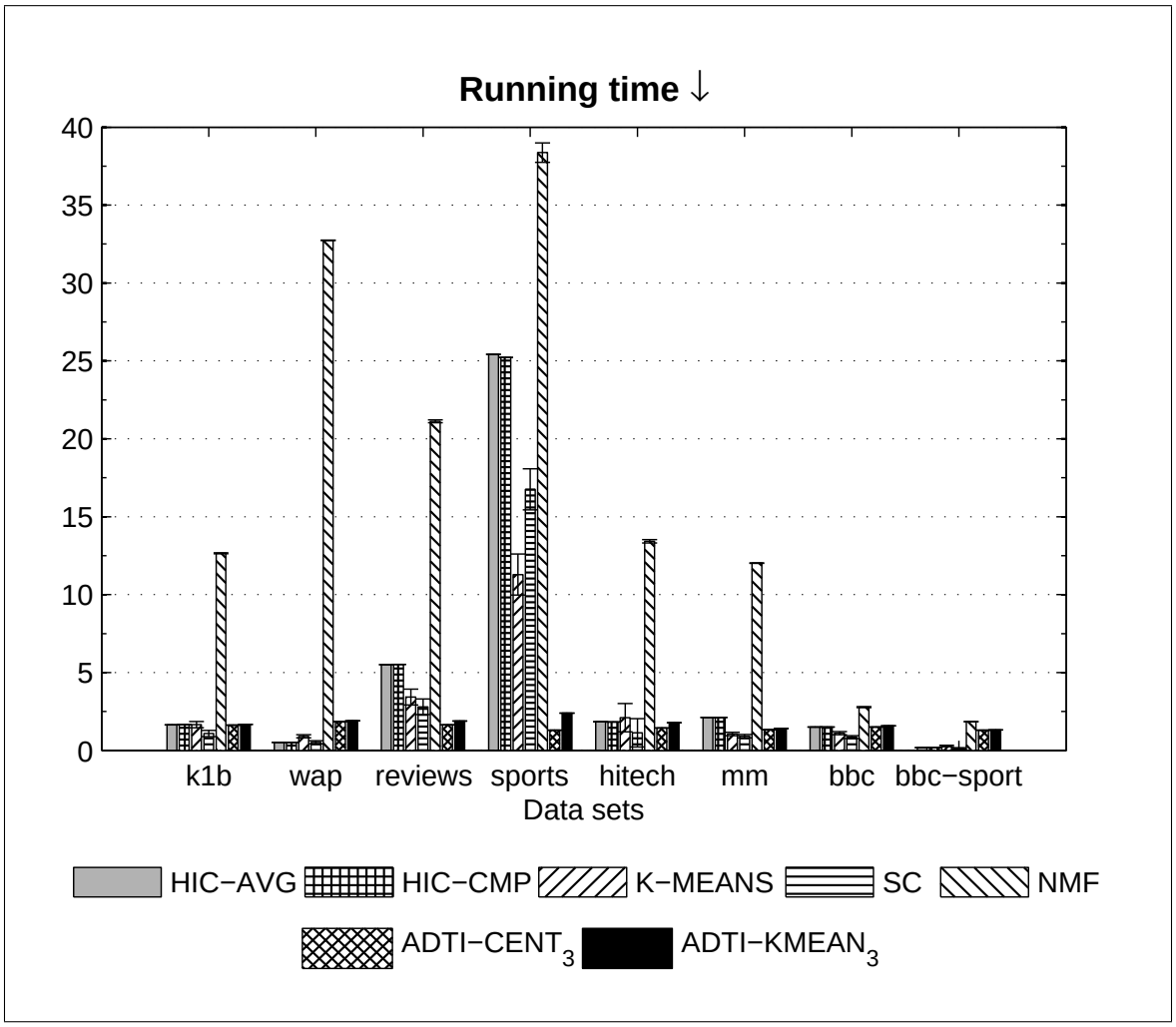

Figure 5.2.7: The output running time of the different document clustering methods and ADTI approaches for 8 data sets.

5.3 Comparing ADTI to Document Classification

In this section, we present the experiments that compare our proposed approach ADTI to document classification. As we discussed in the previous chapter, document classification and ADTI are inequivalent tasks. Hence, in order to make a fair comparison between both tasks, we have conducted two separate experiments. In the first experiment, we wanted to test the effect of using a different data source for training on document classification and compare it with ADTI. The details of this experiment are discussed in section 5.3.1. In the second experiment, we compared ADTI with the ordinary document classification. Although we did not expect that ADTI would perform the same, we wanted to test how much worse the untrained ADTI approach was compared to the trained approach.

Competing Techniques

We compared the performance of two document classification approaches with the proposed ADTI approaches:

- **Nearest Centroid Classifier (*NEAR-CENT*):** We implemented this classifier using MATLAB. In the training phase, the vectors of classes' centroids are calculated using the input document vectors as described in section 2.2.2.1. In the test phase, each document is assigned to the nearest centroid.
- **Support Vector Machines Classifier (*SVM*):** We have used the libsvm implementation to conduct this experiment [120]. As we use libsvm for *document* classification and the document representation has a large number of features, we decided to use the liblinear version of libsvm [121]. As reported by liblinear developers, it has the same performance as the libsvm when used for *document* classification, but it is much more efficient. We have done a parameter tuning on sub-sets of the data sets, and found that the best approach for muli-class SVM is one against all, with a penalty parameter value of 100.
- **ADTI:** The two different variations of the proposed topic identification approaches are used in this comparison: ADTI with nearest centroid identification approach (***ADTI-CENT***$_l$) and ADTI with k-means-based identification approach (***ADTI-KMEAN***$_l$) where l is the number of levels used for the method. We reported here the results of these techniques using different number of levels depending on the experiment. We indicate the number of levels for each experiment separately. For the other ADTI parameters, we used the same parameter values as the last experiments.

Performance Measures

In order to judge the performance of our topic identification method against the document classification approaches, we used the common document classification performance measures. As we reviewed in section 2.2.3, the most common performance measures for document classification are accuracy, F-measure, precision, and recall. The precision, recall, and F-measure of the output is calculated as a weighted average of the per-class values with respect to classes' sizes. In this case, the recall performance measure is equivalent to accuracy. Hence we have reported only F-measure (**Fm**), precision (**P**), and accuracy (**A**). We also measured running time of each technique to compare the efficiency of these approaches.

5.3.1 Comparing ADTI to Document Classification With External Training Data

As we stated earlier, this experiment tested the performance of ADTI against document classification trained from a source other than the input documents. To make the comparison of the two techniques fair, we used the same source that our ontology is created from, Wikipedia, to train the classifiers. As we described in the ADTI methodology, the first step in ADTI is to find the match between the requested topics and the ontology concepts. We know that each concept in our ontology corresponds to a Wikipedia category, so for each data set, we used ADTI topic-concept mapping to gather the whole set of articles that fall under each topic representative category in Wikipedia. These articles are used as the training set for the competing classifiers, then we used input documents as the test set. As we can see, there is no need for multiple training and testing rounds, because the output results for both classifiers and ADTI are deterministic.

In this experiment, we used ADTI approaches without enrichment (ADTI-CENT$_0$ and ADTI-KMEAN$_0$), as the ontology information supplied to our techniques will be the processed version of the documents supplied to the classification techniques. We also reported the results of our approach with 3 levels of enrichment (ADTI-CENT$_3$ and ADTI-KMEAN$_3$) to see the effect of adding more information to our technique compared to the classification techniques.

5.3.1.1 Results and Discussions

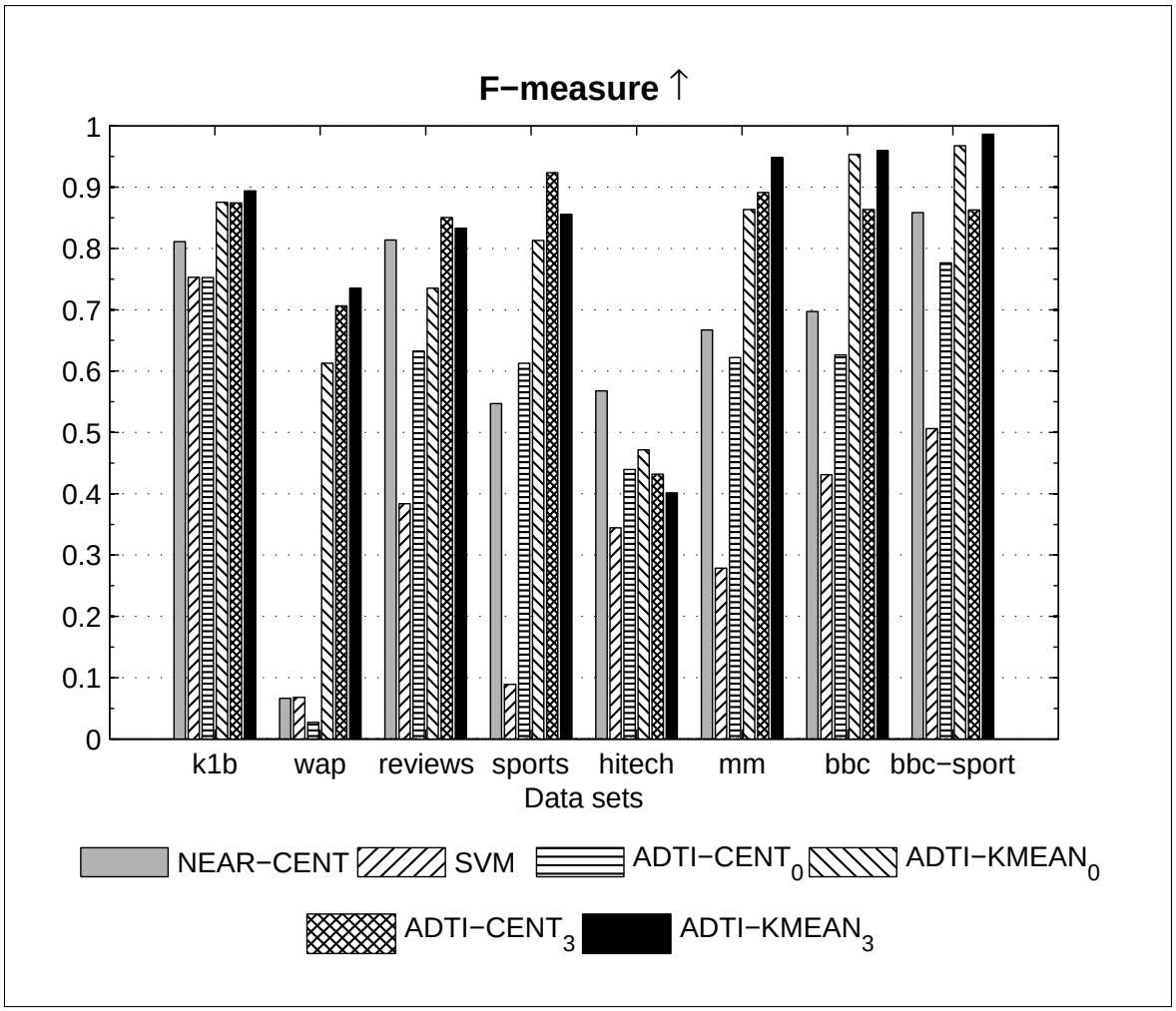

Figure 5.3.1: The output F-measure of the different document classification methods with external training source and ADTI approaches with level 0 and 3 for 8 data sets.

Figures 5.3.1, 5.3.2, and 5.3.3 show the output performance measures using nearest centroid classifier, SVM, and ADTI approaches with 0 and 3 levels of enrichment for eight data sets. Figure 5.3.1 compares the F-measure performance of these methods. From this figure, we can see that ADTI-CENT$_3$ and ADTI-KMEAN$_3$ outperforms both classification techniques in all data sets except in hitech data set where NEAR-CENT classification outperforms all other methods. It can also be observed that ADTI-CENT$_0$ outperforms SVM classification in all data sets except in k1b and wap where they have very close results.

When ADTI-CENT$_0$ is compared with NEAR-CENT classification, we can see that NEAR-CENT classification outperforms ADTI-CENT$_0$ in the reviews and hitech data sets and they have close performance in other data sets. ADTI-KMEAN$_0$ outperforms NEAR-CENT classification in six data sets while they have close results in the reviews data set, and NEAR-CENT classification outperforms ADTI-KMEAN$_0$ in hitech data set.

Regarding accuracy measure results, we can observe from Figure 5.3.3 more or less similar performance as F-measure. Figure 5.3.2 shows precision performance for different approaches. From this figure, it can be seen that ADTI-CENT$_3$ and ADTI-KMEAN$_3$ outperforms both classification techniques in all data sets except in hitech data set where NEAR-CENT classification outperforms all other methods as in F-measure. ADTI-KMEAN$_0$ outperforms NEAR-CENT classification in 6 data sets while they have close results in reviews data set, and NEAR-CENT classification outperforms ADTI-KMEAN$_0$ in hitech data set. ADTI-CENT$_0$ has close performance to NEAR-CENT classification except in reviews and hitech data sets where NEAR-CENT classification outperforms ADTI-CENT$_0$. As for SVM classification, ADTI-KMEAN$_0$ outperforms SVM classification in four data sets while they have close results on other four.

Figure 5.3.4 shows the running time comparison of all the approaches. We can see that NEAR-CENT is the fastest approach, ADTI-CENT$_0$ comes in second, and SVM comes in third.

Table 5.2 shows the overall relative performance measures and running time for the different document classification methods and ADTI approaches. The best output performance is shown in bold and the second best is underlined. This table summarizes the previous results using the approach discussed in section 5.1.4. Table 5.3 shows the statistical significance based on the paired sample one-tailed t-test as described in section 5.2.2. From Table 5.2 and Table 5.3, we can observe that NEAR-CENT classification performance is equivalent to proposed approaches performance for all measure and regardless the number of levels used. We can also notice that SVM classification is significantly inferior to the proposed approaches' performance for all measures, except the ADTI-CENT$_0$ which has equivalent performance to SVM. Regarding the efficiency, we can notice that the proposed approaches are significantly inferior to classification approaches, except in two cases where they are either significantly superior or equivalent to SVM classification.

	$\tilde{Fm}$	$\tilde{P}$	$\tilde{A}$	$\tilde{T}$
NEAR-CENT	0.73±0.29	0.78±0.29	0.73±0.30	**1.00±0.00**
SVM	0.42±0.25	0.63±0.30	0.49±0.27	0.04±0.03
ADTI-CENT$_0$	0.64±0.26	0.71±0.27	0.65±0.25	0.34±0.12
ADTI-KMEAN$_0$	0.91±0.07	0.94±0.07	0.89±0.08	0.07±0.03
ADTI-CENT$_3$	0.93±0.08	0.92±0.07	0.93±0.08	0.00±0.00
ADTI-KMEAN$_3$	**0.95±0.10**	**0.96±0.10**	**0.94±0.12**	0.00±0.00

Table 5.4: The overall relative performance measures for the different document classification methods with external training source and ADTI approaches with level 0 and 3.

M_1	M_2	$\tilde{Fm}$	$\tilde{P}$	$\tilde{A}$	$\tilde{T}$
ADTI-CENT$_0$	NEAR-CENT	$\equiv$	$\equiv$	$\equiv$	$\ll$
	SVM	$\equiv$	$\equiv$	$\equiv$	$\gg$
ADTI-KMEAN$_0$	NEAR-CENT	$\equiv$	$\equiv$	$\equiv$	$\ll$
	SVM	$\gg$	$\gg$	$\gg$	$\equiv$
ADTI-CENT$_3$	NEAR-CENT	$\equiv$	$\equiv$	$\equiv$	$\ll$
	SVM	$\gg$	$\gg$	$\gg$	$\ll$
ADTI-KMEAN$_3$	NEAR-CENT	$\equiv$	$\equiv$	$\equiv$	$\ll$
	SVM	$\gg$	$\gg$	$\gg$	$\ll$

Table 5.5: Comparison between ADTI approaches with level 0 and 3 and different document classification methods with external training source based on statistical significance (using t-test)

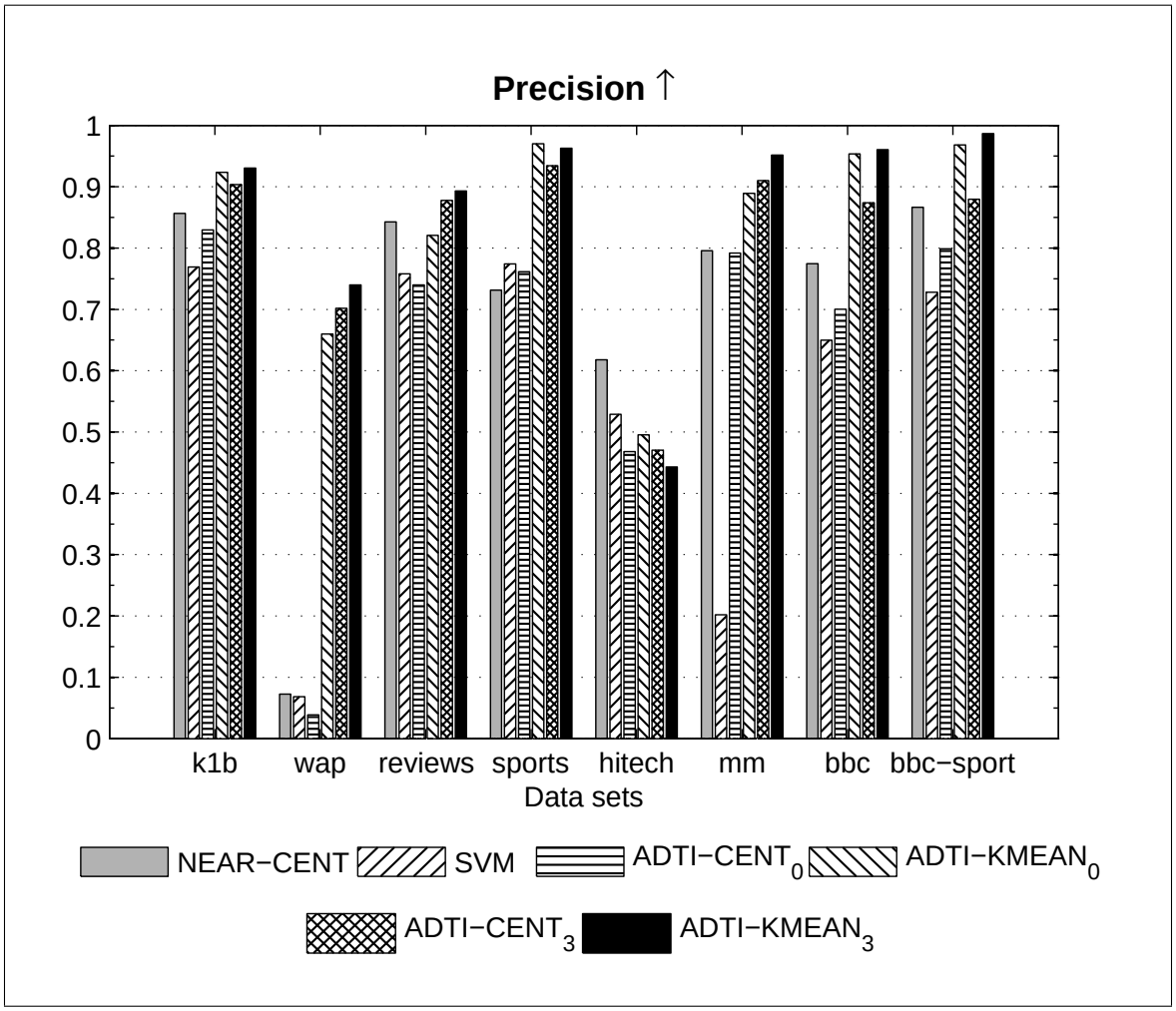

Figure 5.3.2: The output precision of the different document classification methods with external training source and ADTI approaches with level 0 and 3 for 8 data sets.

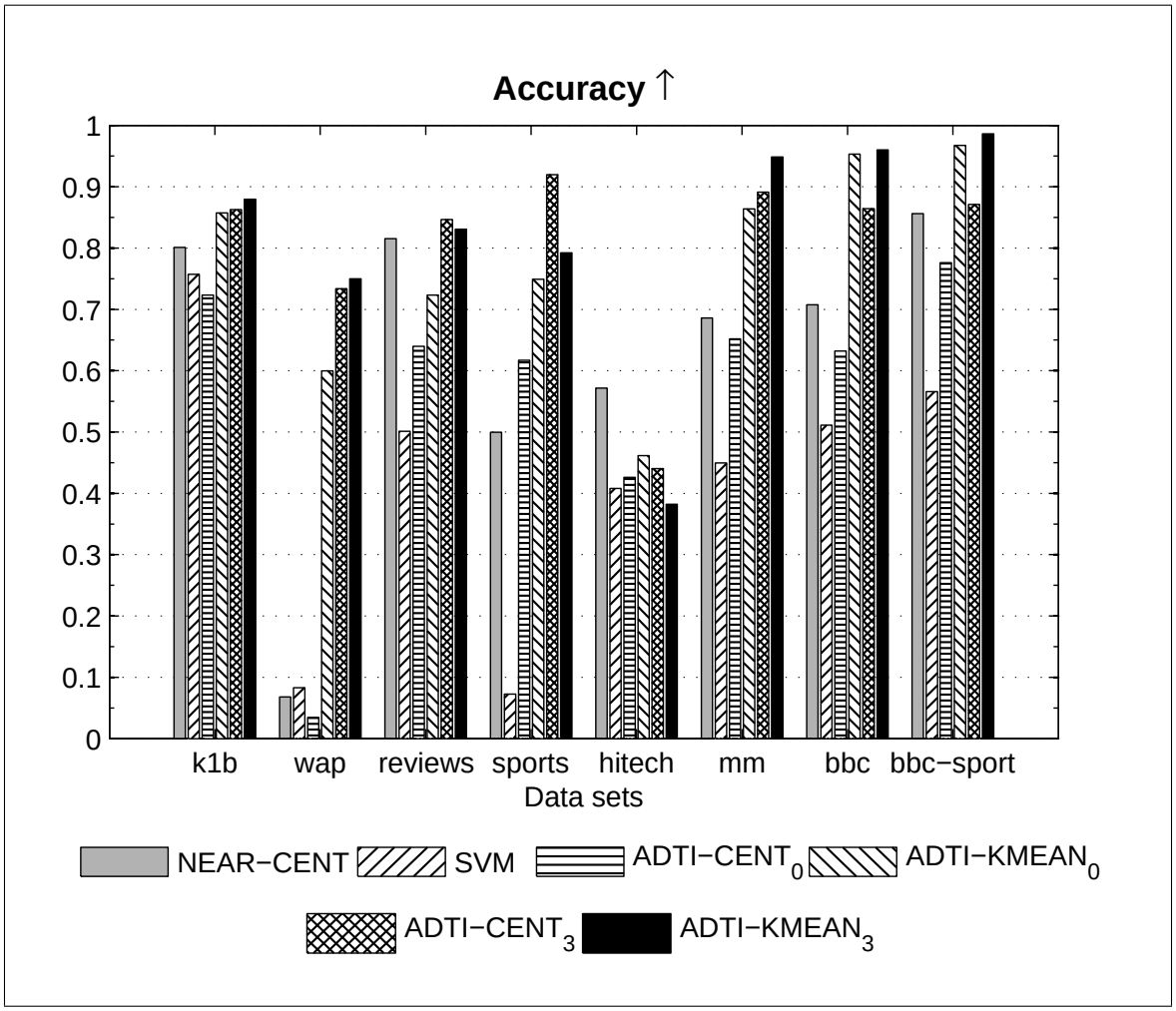

Figure 5.3.3: The output accuracy of the different document classification methods with external training source and ADTI approaches with level 0 and 3 for 8 data sets.

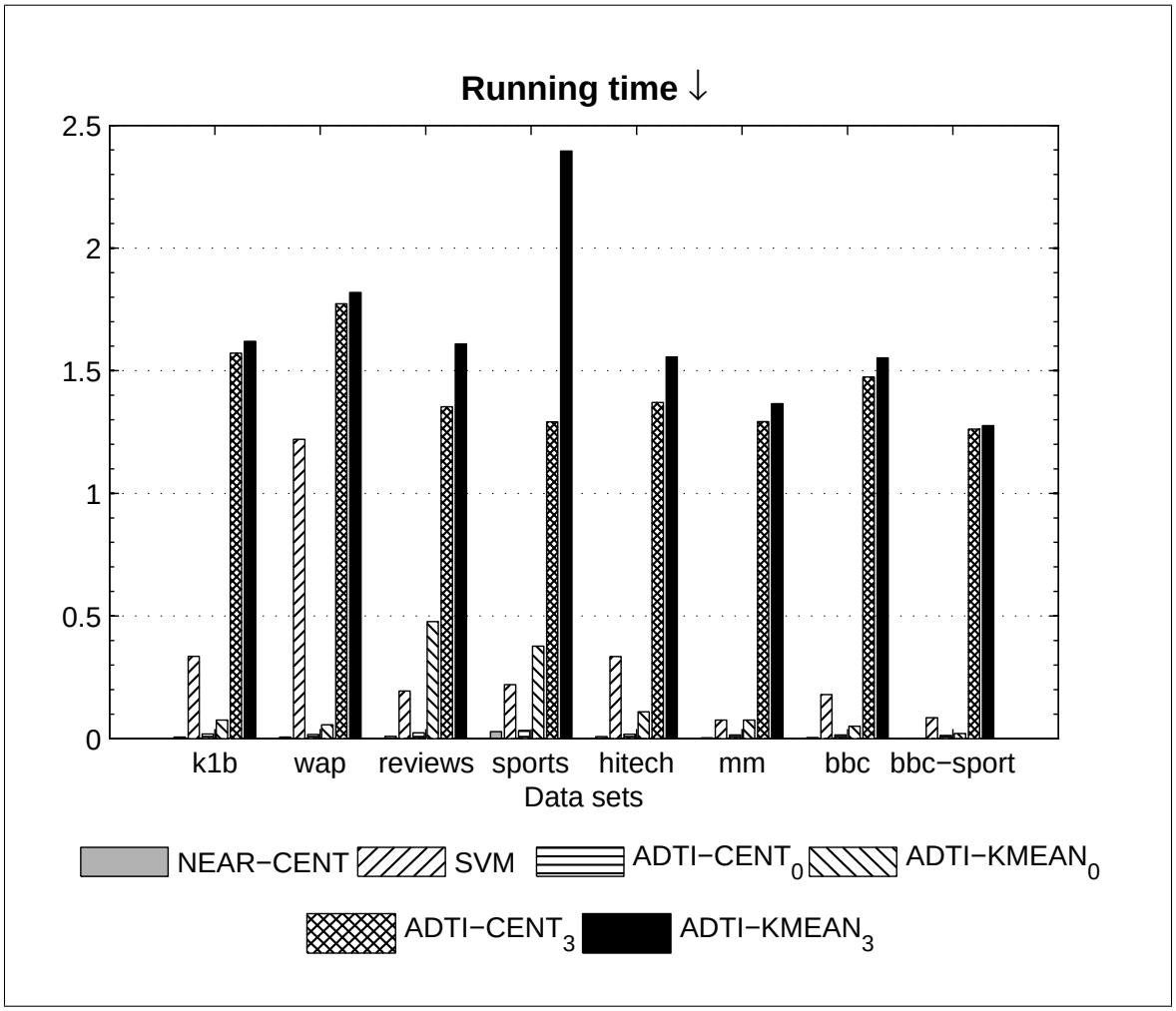

Figure 5.3.4: The output running time of the different document classification methods with external training source and ADTI approaches with level 0 and 3 for 8 data sets.

5.3.2 Comparing ADTI to Ordinary Document Classification

In this experiment, we tested the performance of ADTI against ordinary document classification. What we mean here by "ordinary" is that the classifier is trained from the same document source that we test. This will of course give an advantage to the classifier over our approach, but we want to test if there is a significant difference between ADTI and document classification. We used in our approach three levels of enrichment (ADTI-CENT$_3$ and ADTI-KMEAN$_3$) as in the last two experiments.

We have used the repeated random sub-sampling validation type of the cross-validation technique to compare the performance of the classier. In this approach, we randomly split the input document set into two parts, training and testing parts, in such a way that the training set is 66% of the whole document set and the rest are used for testing. After training the classifiers, we use the test set to check the performance of both ADTI and classifiers. We performed this cross-validation ten times using different partitions, and we report the mean and standard deviation of these rounds.

5.3.2.1 Results and Discussions

Figures 5.3.5, 5.3.6, and 5.3.7 show the output performance measures using nearest centroid classifier, SVM, and ADTI approaches with 0 and 3 levels of enrichment for eight data sets. As we were expecting, classification techniques have better performance in almost all cases. But we can see also that the difference in performance is not large. So, we created Table 5.6 to show the overall relative performance measures and running time for the different document classification methods and ADTI approaches. It summarizes the previous results using the approach discussed in section 5.1.4. We also created Table 5.3 to show the statistical significance of the obtained results based on the paired sample one-tailed t-test as described in section 5.2.2. We can observe that although ADTI-CENT$_3$ performance is significantly inferior to document classification methods in almost all measures, ADTI-KMEAN$_3$ has equivalent performance to SVM classification.

	$\tilde{Fm}$	$\tilde{P}$	$\tilde{A}$	$\tilde{T}$
NEAR-CENT	**0.99±0.01**	**0.99±0.01**	**0.99±0.01**	**1.00±0.00**
SVM	0.94±0.17	0.94±0.17	0.94±0.17	0.05±0.04
ADTI-CENT$_3$	0.85±0.12	0.87±0.11	0.85±0.11	0.03±0.01
ADTI-KMEAN$_3$	0.88±0.12	0.90±0.12	0.87±0.13	0.03±0.01

Table 5.6: The overall relative performance measures for the different document classification methods and ADTI approaches.

M_1	M_2	$\tilde{Fm}$	$\tilde{P}$	$\tilde{A}$	$\tilde{T}$
ADTI-CENT$_3$	NEAR-CENT	$\ll$	$\ll$	$\ll$	$\ll$
	SVM	$\equiv$	$\equiv$	$\equiv$	$\equiv$
ADTI-KMEAN$_3$	NEAR-CENT	$\ll$	$\ll$	$\ll$	$\ll$
	SVM	$\equiv$	$\equiv$	$\equiv$	$\equiv$

Table 5.7: Comparison between ADTI approaches and different document classification methods based on statistical significance (using t-test)

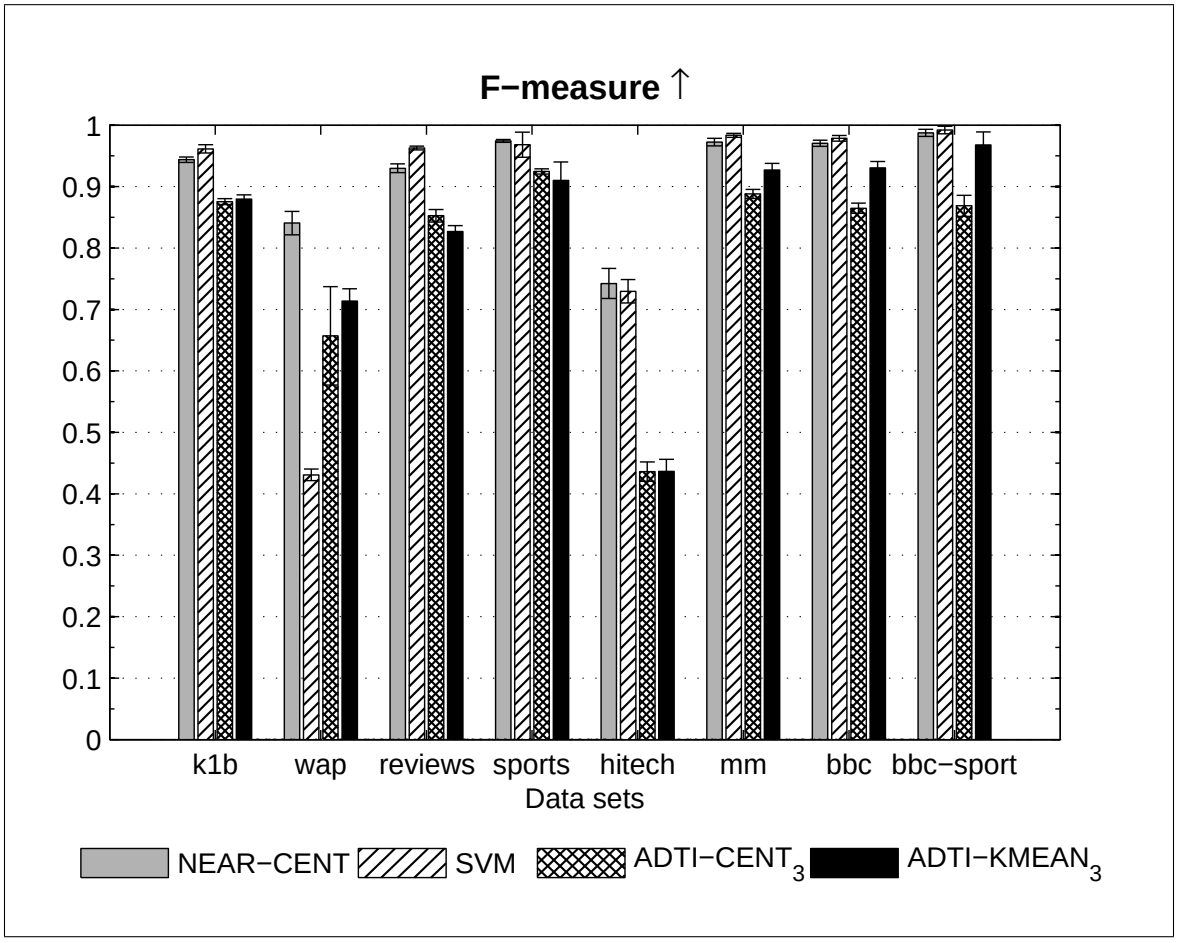

Figure 5.3.5: The output F-measure of the different document classification methods and ADTI approaches for 8 data sets.

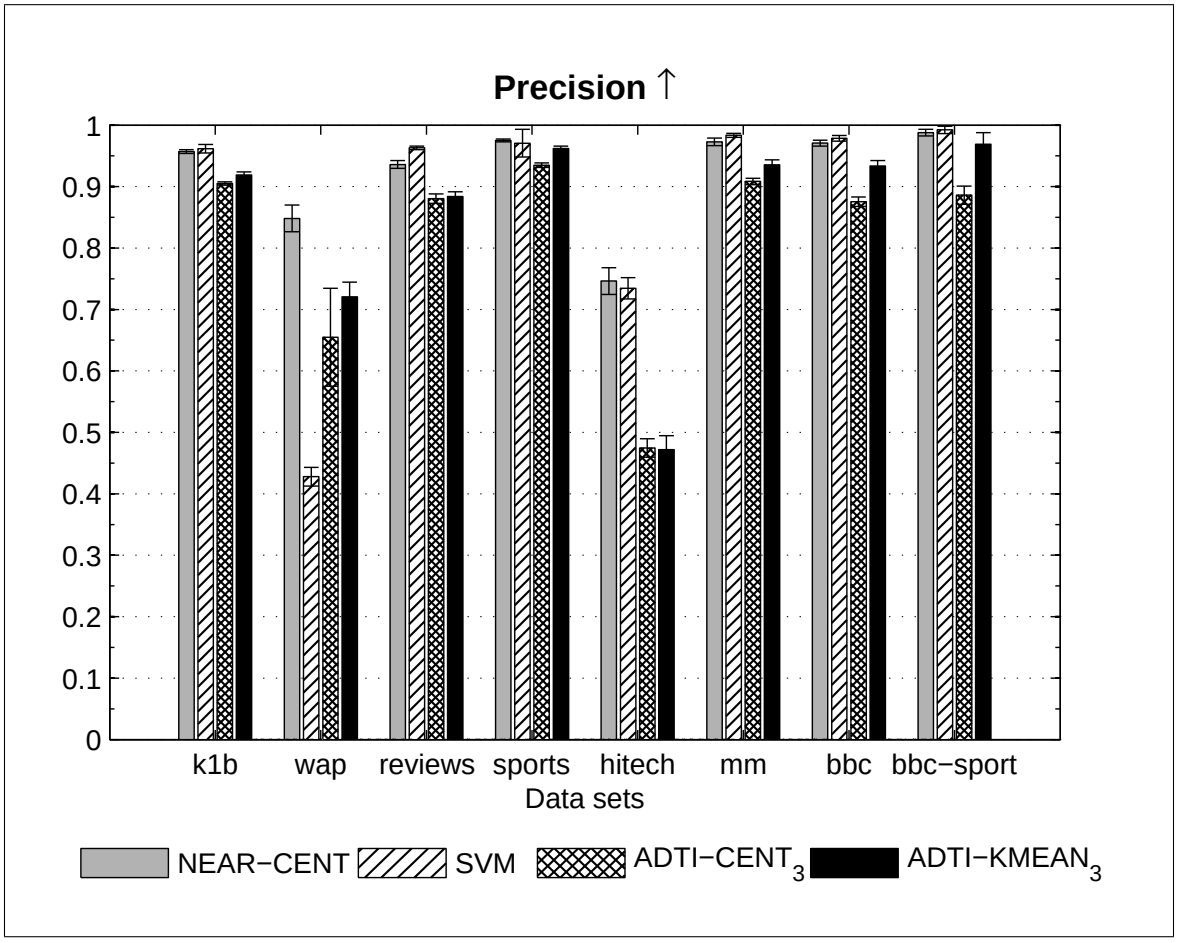

Figure 5.3.6: The output precision of the different document classification methods and ADTI approaches for 8 data sets.

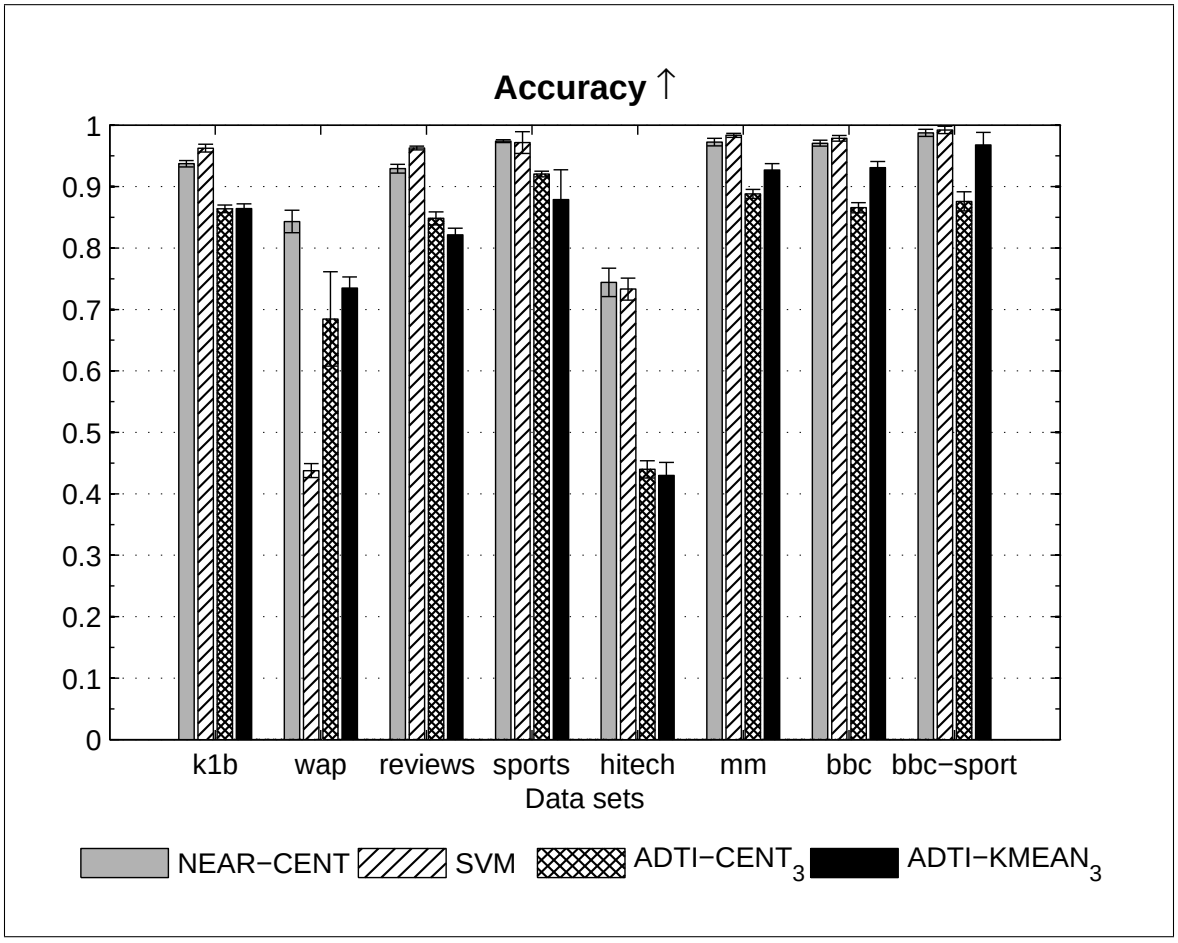

Figure 5.3.7: The output accuracy of the different document classification and ADTI approaches for 8 data sets.

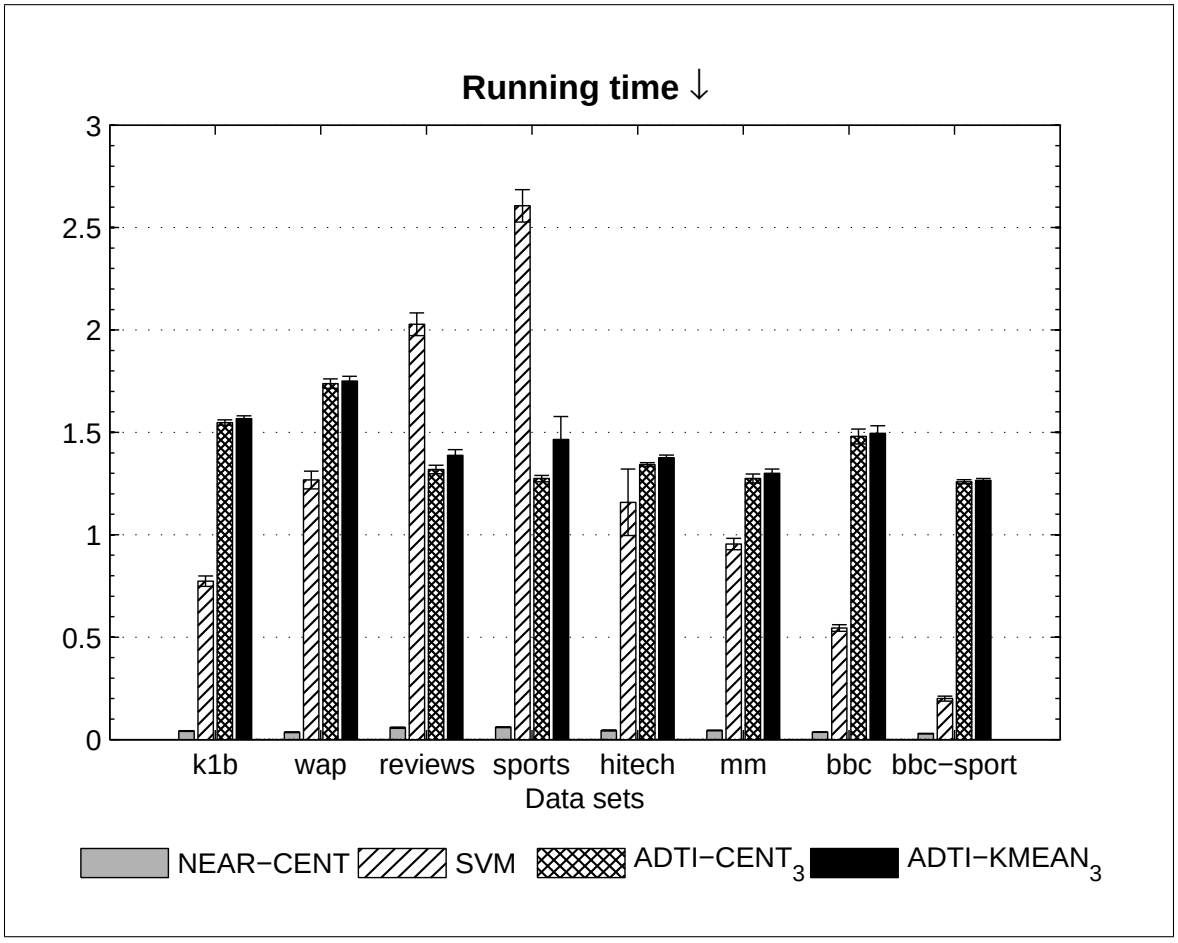

Figure 5.3.8: The output running time of the different document classification methods and ADTI approaches for 8 data sets.

5.4 Summary

In this chapter, we conducted several experiments to test the performance of the proposed ADTI approaches, comparing the performance of ADTI to both document clustering and document classification tasks. In the first experiment, five different clustering approaches with both hierarchical and partitional schema were used. The results show that ADTI outperforms all the tested document clustering approaches. They also showed that ADTI has equivalent efficiency to the document clustering approaches. In the second experiment, we compared ADTI against two document classification approaches with external data for training. We observed that ADTI outperformed SVM classification and achieved an equivalent performance to nearest centroid classification. In the third experiment, we compared ADTI against ordinary document classification. We showed that ADTI has equivalent performance to SVM.

Chapter 6

Conclusion and Future Research

The use of background knowledge to enhance the performance of text mining has been proposed and widely used in different applications. This work proposes yet another approach for extracting, organizing, and utilizing the background knowledge in the form of an ontology. This ontology can be used to improve the performance of different text mining tasks. We introduce a novel task in text mining for topic indexing. We utilize the knowledge extracted in the ontology to automatically identify document topics. We call this task the automatic document topic identification, ADTI.

In this chapter, we summarize the main contributions we have proposed and achievements that we have attained so far. This is followed by the future plan for this research.

6.1 Proposed Framework and Achievements

In our framework, to benefit from the background knowledge and to use it to enrich the text mining task, we have defined the following main modules. The first module is concerned with how to build an organized and structured form of knowledge, ontology, from a different format of knowledge repository. The second module details how to utilize this knowledge structure, an ontology, for a newly defined task, ADTI. In the remainder of this section, we summarize how the proposed framework defines these three modules, then we show how we applied this framework to Wikipedia.

- **Building the ontology:** We have proposed the mechanism needed to build the ontology used for text mining tasks. We have discussed how to extract the concepts structural relations which is used to construct the the ontology taxonomy. We also

proposed how to extract nonstructural relations between the concepts in a form of semantic relatedness between them. This is done by utilizing the representative articles for each concept. After preprocessing these articles, we showed how to select and weight the terms from these articles to represent concepts. Hence, they can be used to evaluate the semantic relatedness between concepts.

- **Using Wikipedia as a knowledge repository:** We have investigated the reasons for selecting Wikipedia as a knowledge repository. The various difficulties and the implementation issues have been discussed in detail in section 3.3. Therefore, we have used Wikipedia to create the Wikipedia Hierarchical Ontology, WHO.

- **Automatic document topic identification using WHO:** We have applied the proposed ontology to automatically identify documents' topics. We use a specific topics to find the best match one for each input document. We have shown in chapter 4 the setups that we used for this application.

  We have obtained results with high values for the performance metrics. We have also observed that although the document identification task is very similar to the document clustering task in the sense that they are both unsupervised learning (the input documents are not used for training), the performance of the document identification task is much better. This is a consequence of two major causes.

  First, identifying the topic of interest simplifies the problem of grouping similar documents. Instead of trying to match the documents blindly, it matches them according to a specific set of topics. In most real-world cases, we know the topics of interest and want to group similar documents according to these topics. Second, using well-formatted background knowledge helps to find the similarity between that topic and the documents, and hence tends to group the most similar documents together.

  We also compared ADTI against document classification. We showed that ADTI outperforms one of the state-of-the-art methods for document classification [13], SVM, when it is trained from the same source as ADTI (Wikipedia) and ADTI achieved equivalent performance with SVM when it is trained normally. This makes ADTI superior to SVM, as ADTI does not need to be trained with every new input data set.

6.2 Contributions

This thesis makes the following research contributions:

6.2.1 Survey of the Task

In the literature, there are many highly related research problems to our proposed work. Chapter 2 surveys the different text mining tasks that are related to our proposed work. It also surveys the task of ontology creation along with the different features that differentiate between different ontologies. It also reviews the state-of-the-art methods and techniques that employ the use of the background knowledge in text mining.

6.2.2 New Techniques

Finding the most appropriate topic for a document is a difficult task. There are many approaches in text mining that are concerned with this goal. Some of these approaches are unsupervised, such as document clustering with cluster labeling, and some of them are supervised, such as document classification. Also there are some other techniques that accomplish the task with the help of background knowledge, such as document tagging. In this dissertation, we propose a novel semi-supervised learning approach to automatically identify document topics. In this approach, we utilize human background knowledge to select the most relevant topic for input documents. What makes this approach a semi-supervised learning one is that we take a list of required topics as an input to the approach. This also introduces a restriction to this approach as the output assignment should only be from this list.

We also showed how to extract the human background knowledge to a machine-usable form, namely ontology. The format of the knowledge represented in the ontology is generic and hence it can be used for other text mining tasks. As an example of another application of the ontology, we showed how we can utilize this ontology to improve the document representation. Although the proposed ontology extraction approach is generic and can be applied on any knowledge repository that meets some requirements, we showed how to apply this approach to Wikipedia.

The list of contributions can be summarized as follows:

- Designed and introduced a new ontology structure that can store the human background knowledge in a form that can be used for text mining tasks.
- Proposed and developed a new technique for extracting the knowledge from any knowledge repository to our proposed ontology.
- Applied the ontology extraction approach to Wikipedia to create a Wikipedia Hierarchical Ontology, WHO.

- Defined a new task in text mining that takes a list of topics of interests and assign a single topic to each input documents using only background knowledge. We called this task Automatic Document Topic Identification, ADTI.
- Proposed and developed two new techniques that utilize the proposed ontology structure in the newly defined task, ADTI.

6.3 Future Work

We have proposed many ideas and techniques for building the WHO and for employing it in document modeling and in different applications in text mining. This work can be extended to be utilized in different applications. Following are some thoughts as to how this work could be expanded.

Tree-based Document Clustering. We have presented the document clustering using the standard similarity measure after modeling documents with the ontology. We can apply this task using a different similarity measure based on the document ontology taxonomy. After mapping the document to WHO concepts, we can evaluate different similarity measures which could be calculated, based on the paths that connect the concepts of each document.

Cluster Topic Prediction. Cluster labeling is the process of selecting a representative label (topic) for each obtained cluster from document clustering process. The source of these labels is usually from the terms and/or the phrases which the input documents are indexed with. Instead of using documents' frequent terms for labels, we can use the ontology concepts to label clusters. In contrast with document terms, ontology concepts are usually representative of topics. Also, concepts have more mature features defined in the ontology, in the form of a taxonomy and semantic relatedness. So these features can be used to identify the most appropriate label for each cluster.

Use a Different Knowledge Repository. So far, all the reported performance and results are based on using WHO, which is created from Wikipedia. Our ontology extraction approach, proposed in chapter 3, is generic and can be applied to different knowledge repositories which meet some specifications. Hence, our approach can be applied to different knowledge repositories, such as the Open Directory Project (ODP), Google Web

Directory, or Yahoo! Web Directory, with examination of how this affects the performance of the system.

Applying Proposed Ontology to Different Text Mining Tasks. We have examined the use of an ontology for ADTI. We can examine the effect of mapping documents to WHO on text classification, and how it will improve the output performance with respect to VSM. Also we can use the ontology for document tagging. This task is different from ADTI in that the topics for the document are not known beforehand. We can assign to each document the most relevant concepts (tags) to its contents.

Applying ADTI on Big Data. Recently, many different techniques and algorithms have been developed to deal with big data. Big data usually refers to data sets with sizes that are beyond the normal storage size, and which need to be split to many storage units, machines, to be able to store it. The common techniques for handling the data and deriving patterns and trends from it are not applicable for this type of data set as the data is distributed over multiple machines. One of the common frameworks that is used to develop efficient algorithms that can handle big data is the MapReduce framework [122]. One of the proposed extension of ADTI is to develop a good implementation of it over the MapReduce framework. By doing so, our algorithm will be applicable on huge data sets that are distributed over thousands machines.

Measuring Knowledge Repository Growing Effect. We have used Wikipedia as our source of knowledge to create our ontology. One of the features of Wikipedia is that it is continuously growing with time. One of the extension ideas for our proposed work is to test the effect of the increase of the knowledge on the ontology and hence on ADTI application. This test can be done by using a small subset of Wikipedia articles and categories to create the ontology and test it on ADTI. Then, we can increase this subset gradually and test how this increase affects the quality of the topics' assignments. We can also check if there is a certain threshold after which an increase in information would not increase the performance.

Testing ADTI with Multilabel Assignment. As we mentioned in our proposed approach, ADTI is used to assign each of the input documents to one of the input topics. Sometimes input documents may belong to multiple topics. For example, an input document may cover the effect of some political actions on the economics. In this case, this document may fall under both the "Economics" topic and the "Politics" topic. So, we can

use our proposed approach to assign each document to more than one topic from a pool of topics based on the similarity measure as we proposed. Thus we propose, as an extension to ADTI, a multilabel assignment test to check the performance of ADTI if it were used to assign to each document more than one label.

Permissions

Parts of Chapters 3 and 4 are reprinted from the following papers.

M. M. Hassan, F. Karray, and M. S. Kamel, "Automatic Document Topic Identification using Wikipedia Hierarchical Ontology," in Proceedings of the Eleventh IEEE International Conference on Information Science, Signal Processing and their Applications (ISSPA'12), 2012, pp. 237-242.

References

[1] M. Uschold and M. Gruninger, "Ontologies and semantics for seamless connectivity," *ACM SIGMod Record*, vol. 33, no. 4, pp. 58–64, 2004.

[2] M. Grobelnik, "Text mining for ontology learning," SEKT 1st Advisory Board Meeting, 2004.

[3] K. Coursey and R. Mihalcea, "Topic identification using Wikipedia graph centrality," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*. Association for Computational Linguistics, 2009, pp. 117–120.

[4] O. Medelyan, I. Witten, and D. Milne, "Topic indexing with Wikipedia," in *Proceedings of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, 2008, pp. 19–24.

[5] R. Feldman and I. Dagan, "Knowledge discovery in textual databases (KDT)," in *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, 1995, pp. 112–117.

[6] G. Salton, A. Wong, and C. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, pp. 613 – 620, November 1975.

[7] S. Wong, W. Ziarko, and P. Wong, "Generalized vector spaces model in information retrieval," in *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM New York, NY, USA, 1985, pp. 18–25.

[8] M. Keikha, N. Razavian, F. Oroumchian, and H. Razi, "Document Representation and Quality of Text: An Analysis," *Survey of Text Mining II: Clustering, Classification, and Retrieval*, pp. 219–232, 2007.

[9] K. M. Hammouda and M. S. Kamel, "Phrase-based document similarity based on an index graph model," in *ICDM*, 2002, pp. 203–210.

[10] ——, "Document similarity using a phrase indexing graph model," *Knowl. Inf. Syst.*, vol. 6, no. 6, pp. 710–727, 2004.

[11] ——, "Efficient phrase-based document indexing for web document clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 10, pp. 1279–1296, 2004.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995, 10.1007/BF00994018.

[13] J. D. M. Rennie and R. Rifkin, "Improving multiclass text classification with the support vector machine," Artificial Intelligence Lab, MIT, Tech. Rep., 2001.

[14] K. Duan and S. Keerthi, "Which is the best multiclass svm method? an empirical study," *Multiple Classifier Systems*, pp. 732–760, 2005.

[15] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *Neural Networks, IEEE Transactions on*, vol. 13, no. 2, pp. 415 –425, mar 2002.

[16] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," *Advances in neural information processing systems*, vol. 12, no. 3, pp. 547–553, 2000.

[17] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *CoRR*, vol. cs.AI/9501101, 1995.

[18] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *The Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2002.

[19] E.-H. Han and G. Karypis, "Centroid-based document classification: Analysis and experimental results," in *Principles of Data Mining and Knowledge Discovery*, ser. Lecture Notes in Computer Science, D. Zighed, J. Komorowski, and J. Zytkow, Eds. Springer Berlin / Heidelberg, 2000, vol. 1910, pp. 116–123.

[20] T. Hofmann, "The Cluster-Abstraction Model: Unsupervised Learning of Topic Hierarchies from Text Data," in *International Joint Conference on Artificial Intelligence*, vol. 16, 1999, pp. 682–687.

[21] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen, "Websom - self-organizing maps of document collections," *Neurocomputing*, vol. 21, no. 1-3, pp. 101 – 117, 1998.

[22] L. Kaufman, P. Rousseeuw, D. D. o. M. Technische Hogeschool, and Informatics., "Clustering by means of medoids," Technische Hogeschool, Delft(Netherlands). Dept. of Mathematics and Informatics., Tech. Rep., 1987.

[23] J. Bezdek and R. Ehrlich, "FCM: The fuzzy c-means clustering algorithm." *Comp. Geosci.*, vol. 10, no. 2, pp. 191–203, 1984.

[24] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *KDD workshop on text mining*, vol. 34, 2000, p. 35.

[25] I. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means, spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM New York, NY, USA, 2004, pp. 551–556.

[26] I. Dhillon and D. Modha, "Concept decompositions for large sparse text data using clustering," *Machine Learning*, vol. 42, no. 1, pp. 143–175, 2001.

[27] E. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," *Biometrics*, vol. 21, pp. 768–769, 1965.

[28] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 281-297, 1967, p. 14.

[29] J. Tou and R. Gonzalez, "Pattern recognition principles," *Image Rochester NY*, vol. 7, 1974.

[30] L. Kaufman, P. Rousseeuw, and E. Corporation, *Finding groups in data: an introduction to cluster analysis*. Wiley Online Library, 1990, vol. 39.

[31] J. Pena, J. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern recognition letters*, vol. 20, no. 10, pp. 1027–1040, 1999.

[32] I. Katsavounidis, J. Kuo, and Z. Zhang, "A new initialization technique for generalized lloyd iteration," *Signal Processing Letters, IEEE*, vol. 1, no. 10, pp. 144–146, 1994.

[33] P. Bradley and U. Fayyad, "Refining initial points for k-means clustering," in *Proceedings of the Fifteenth International Conference on Machine Learning*, vol. 66. Citeseer, 1998.

[34] S. Redmond and C. Heneghan, "A method for initialising the k-means clustering algorithm using kd-trees," *Pattern Recognition Letters*, vol. 28, no. 8, pp. 965–973, 2007.

[35] H. Suo, K. Nie, X. Sun, and Y. Wang, "One optimized choosing method of k-means document clustering center," *Information Retrieval Technology*, pp. 490–495, 2008.

[36] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[37] M. Maila and J. Shi, "A random walks view of spectral segmentation," in *AI and STATISTICS (AISTATS) 2001*, 2001.

[38] A. Ng, M. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[39] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[40] ——, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, pp. 556–562, 2001.

[41] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003, pp. 267–273.

[42] W. Xu and Y. Gong, "Document clustering by concept factorization," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2004, pp. 202–209.

[43] O. Medelyan, "Human-competitive automatic topic indexing," Ph.D. dissertation, The University of Waikato, 2009.

[44] F. Lancaster, *Indexing and abstracting in theory and practice*. Facet, 2003.

[45] A. Popescul and L. H. Ungar, "Automatic labeling of document clusters," 2000.

[46] K. Coursey, R. Mihalcea, and W. Moen, "Using encyclopedic knowledge for automatic topic identification," in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2009, pp. 210–218.

[47] P. Schönhofen, "Identifying document topics using the Wikipedia category network," *Web Intelligence and Agent Systems*, vol. 7, no. 2, pp. 195–207, 2009.

[48] D. Huynh, T. Cao, P. Pham, and T. Hoang, "Using Hyperlink Texts to Improve Quality of Identifying Document Topics Based on Wikipedia," in *Knowledge and Systems Engineering, 2009. KSE'09. International Conference on*. IEEE, 2009, pp. 249–254.

[49] M. Janik and K. Kochut, "Training-less ontology-based text categorization," in *ECIR workshop on exploiting semantic annotations in information retrieval (ESAIR'08)*, 2008.

[50] ——, "Wikipedia in action: Ontological knowledge in text categorization," in *Semantic Computing, 2008 IEEE International Conference on*. IEEE, 2008, pp. 268–275.

[51] S. Auer and J. Lehmann, "What have innsbruck and leipzig in common? extracting semantics from wiki content," in *The Semantic Web: Research and Applications*. Springer, 2007, pp. 503–517.

[52] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics (NRL)*, vol. 52, no. 1, pp. 7–21, 2005.

[53] C. J. van Rijsbergen, *Information Retrieval*, 2nd ed. Butterworths, 1979.

[54] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–199, 1993.

[55] T. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human Computer Studies*, vol. 43, no. 5, pp. 907–928, 1995.

[56] N. Guarino and P. Giaretta, "Ontologies and knowledge bases: Towards a terminological clarification," *Towards very large knowledge bases: knowledge building and knowledge sharing*, vol. 1, no. 9, p. 9, 1995.

[57] L. Garshol, "Metadata? Thesauri? Taxonomies? Topic maps! Making sense of it all," *Journal of Information Science*, vol. 30, no. 4, p. 378, 2004.

[58] W. Siricharoen, "Ontologies and object models in object oriented software engineering," *IAENG International Journal of Computer Science*, vol. 33, no. 1, pp. 19–24, 2007.

[59] N. Noy, D. McGuinness *et al.*, "Ontology development 101: A guide to creating your first ontology," 2001.

[60] M. Boulos, "Semantic wikis: A comprehensible introduction with examples from the health sciences," *Journal of Emerging Technologies in Web Intelligence*, vol. 1, no. 1, pp. 94–96, 2009.

[61] I. Niles and A. Pease, "Origins of the ieee standard upper ontology," in *Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology*. Citeseer, 2001, pp. 4–10.

[62] J. Blake, C. Bult *et al.*, "Beyond the data deluge: data integration and bio-ontologies," *Journal of biomedical informatics*, vol. 39, no. 3, pp. 314–320, 2006.

[63] R. Beckwith and G. Miller, "Implementing a lexical network," *International Journal of Lexicography*, vol. 3, no. 4, pp. 302–312, 1990.

[64] D. Pisanelli, "Biodynamic ontology: applying bfo in the biomedical domain," *Ontologies in medicine*, vol. 102, p. 20, 2004.

[65] A. Osterwalder, Y. Pigneur *et al.*, "An e-business model ontology for modeling e-business," in *15th Bled Electronic Commerce Conference*. Bled, Slovenia, 2002, pp. 17–19.

[66] D. B. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd, "Cyc: toward programs with common sense," *Commun. ACM*, vol. 33, no. 8, pp. 30–49, 1990.

[67] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. Goldberg, K. Eilbeck, A. Ireland, C. Mungall *et al.*, "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration," *Nature biotechnology*, vol. 25, no. 11, pp. 1251–1255, 2007.

[68] J. Brank, M. Grobelnik, and D. Mladenić, "A survey of ontology evaluation techniques," in *In In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD)*, 2005.

[69] A. Maedche and S. Staab, "Measuring similarity between ontologies," in *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, ser. Lecture Notes in Computer Science, A. Gómez-Pérez and V. Benjamins, Eds. Springer Berlin / Heidelberg, 2002, vol. 2473, pp. 15–21.

[70] R. Porzel and R. Malaka, "A task-based approach for ontology evaluation," in *ECAI Workshop on Ontology Learning and Population, Valencia, Spain*. Citeseer, 2004.

[71] C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks, "Data driven ontology evaluation," in *International Conference on Language Resources and Evaluation (LREC 2004)*, 2004.

[72] A. Lozano-Tello and A. Gómez-Pérez, "Ontometric: A method to choose the appropriate ontology," *Journal of Database Management*, vol. 2, no. 15, pp. 1–18, 2004.

[73] R. Mihalcea and D. Moldovan, "Semantic indexing using WordNet senses," in *Proceedings of ACL Workshop on IR & NLP*, 2000, pp. 35–45.

[74] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007, pp. 6–12.

[75] N. Aussenac-Gilles and J. Mothe, "Ontologies as background knowledge to explore document collections," in *Actes de la Conférence sur la Recherche d'Information Assistée par Ordinateur (RIAO)*, 2004, pp. 129–142.

[76] A. Maedche and S. Staab, "Ontology learning," in *Handbook on ontologies*. Springer Verlag, 2004, pp. 173–190.

[77] N. Aussenac-Gilles, B. Biébow, and S. Szulman, "Revisiting ontology design: A method based on corpus analysis," in *12th Int. Conference in Knowlegde Engineering and knowledge Management*, vol. 1937. Springer, 2000, pp. 172–188.

[78] J. Mothe, C. Chrisment, B. Dousset, and J. Alaux, "DocCube: multi-dimensional visualisation and exploration of large document sets," *Journal of the American Society for Information Science and Technology*, vol. 54, no. 7, pp. 650–659, 2003.

[79] S. Shehata, F. Karray, and M. Kamel, "A concept-based model for enhancing text categorization," in *KDD*, 2007, pp. 629–637.

[80] B. Choudhary and P. Bhattacharyya, "Text clustering using semantics," in *Proceedings of the 11th International World Wide Web Conference*, 2002.

[81] H. Uchida, M. Zhu, and S. T.D., "Unl: A gift for a millennium." The United Nations University, Tech. Rep., 1995.

[82] G. Miller, "WordNet: a lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[83] E. Christiane Fellbaum, *WordNet. An electronic lexical database*. preface by George Miller. Cambridge, MA: MIT Press, 1998.

[84] A. Hotho, S. Staab, and G. Stumme, "Ontologies improve text document clustering," in *Third IEEE International Conference on Data Mining, 2003. ICDM 2003*, 2003, pp. 541–544.

[85] ——, "Text clustering based on background knowledge," University of Karlsruhe, Institute AIFB, Tech. Rep., 2003.

[86] ——, "Wordnet improves text document clustering," in *Proc. of the SIGIR 2003 Semantic Web Workshop*, 2003, pp. 541–544.

[87] E. Agirre and G. Rigau, "Word sense disambiguation using conceptual density," in *Proceedings of COLING*, vol. 96, 1996, pp. 16–22.

[88] D. Lewis, "Reuters-21578 text categorization test collection," AT&T Research Lab, Tech. Rep., 1997.

[89] J. Sedding and D. Kazakov, "Wordnet-based text document clustering," in *COLING 2004 3rd Workshop on Robust Methods in Analysis of Natural Language Data*, V. Pallotta and A. Todirascu, Eds. Geneva, Switzerland: COLING, August 29th 2004, pp. 104–113.

[90] W. K. Gad and M. S. Kamel, "Enhancing Text Clustering Performance Using Semantic Similarity," in *Enterprise Information Systems: 11th International Conference, ICEIS 2009, Milan, Italy, May 6-10, 2009, Proceedings*. Springer, 2009, pp. 325–335.

[91] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone," in *Proceedings of the 5th annual international conference on Systems documentation*. ACM New York, NY, USA, 1986, pp. 24–26.

[92] J. Giles, "Internet encyclopaedias go head to head," *Nature*, vol. 438, no. 7070, pp. 900–901, 2005.

[93] M. Strube and S. Ponzetto, "WikiRelate! Computing semantic relatedness using Wikipedia," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 21, no. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, p. 1419.

[94] Z. Syed, T. Finin, and A. Joshi, "Wikipedia as an ontology for describing documents," in *Proceedings of the Second International Conference on Weblogs and Social Media. AAAI Press*, 2008.

[95] P. Wang, J. Hu, H. Zeng, and Z. Chen, "Using wikipedia knowledge to improve text classification," *Knowledge and Information Systems*, vol. 19, no. 3, pp. 265–281, 2009.

[96] E. Gabrilovich and S. Markovitch, "Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge," in *Proceedings of the National Conference On Artificial Intelligence*, vol. 21, no. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, p. 1301.

[97] S. Banerjee, "Boosting inductive transfer for text classification using wikipedia," in *Proceedings of the 6th International Conference on Machine Learning and Applications (ICMLA)*, 2007, pp. 148–153.

[98] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi, "Learning to classify short and sparse text & web with hidden topics from large-scale data collections," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*. New York, NY, USA: ACM, 2008, pp. 91–100.

[99] S. Banerjee, K. Ramanathan, and A. Gupta, "Clustering short texts using Wikipedia," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM New York, NY, USA, 2007, pp. 787–788.

[100] J. Hu, L. Fang, Y. Cao, H. Zeng, H. Li, Q. Yang, and Z. Chen, "Enhancing text clustering by leveraging Wikipedia semantics," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM New York, NY, USA, 2008, pp. 179–186.

[101] X. Hu, X. Zhang, C. Lu, E. K. Park, and X. Zhou, "Exploiting Wikipedia as external knowledge for document clustering," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM New York, NY, USA. ACM, 2009, pp. 389–396.

[102] A. Huang, D. Milne, E. Frank, and I. Witten, "Clustering Documents with Active Learning using Wikipedia," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. IEEE Computer Society Washington, DC, USA, 2008, pp. 839–844.

[103] ——, "Clustering Documents Using a Wikipedia-Based Concept Representation," in *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. Springer, 2009, pp. 628–636.

[104] S. Ponzetto and M. Strube, "Deriving a large scale taxonomy from Wikipedia," in *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, vol. 22, no. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007, p. 1440.

[105] Y. Ding and R. Engels, "Ir and ai: Using co-occurrence theory to generate lightweight ontologies," in *Database and Expert Systems Applications, 2001. Proceedings. 12th International Workshop on*. IEEE, 2001, pp. 961–965.

[106] A. Muller, J. Dorre, P. Gerstl, and R. Seiffert, "The TaxGen framework: automating the generation of a taxonomy for alarge document collection," in *System Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on*, 1999.

[107] D. Younger, "Minimum feedback arc sets for a directed graph," *Circuit Theory, IEEE Transactions on*, vol. 10, no. 2, pp. 238–245, 1963.

[108] P. Festa, P. Pardalos, M. Resende *et al.*, "Feedback set problems," *Handbook of combinatorial optimization*, vol. 4, pp. 209–258, 1999.

[109] B. Berger and P. W. Shor, "Approximation alogorithms for the maximum acyclic subgraph problem," in *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '90. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1990, pp. 236–243.

[110] K. H. F. Brandenburg, "Sorting Heuristics for the Feedback Arc Set Problem," 2011, no. MIP-1104.

[111] P. Eades, X. Lin, and W. Smyth, "A fast and effective heuristic for the feedback arc set problem," *Information Processing Letters*, vol. 47, no. 6, pp. 319 – 323, 1993.

[112] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[113] M. Hassan, F. Karray, and M. S. Kamel, "Improving document clustering using a hierarchical ontology extracted from wikipedia," in *SIAM SDM Text Mining Workshop*. Columbus, OH, USA: SIAM, May 2010.

[114] Y. Zhao, G. Karypis, and U. Fayyad, "Hierarchical clustering algorithms for document datasets," *Data Mining and Knowledge Discovery*, vol. 10, no. 2, pp. 141–168, 2005.

[115] E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore, "WebACE: A web agent for document categorization and exploration," in *Proceedings of the second international conference on Autonomous agents*. ACM, 1998, pp. 408–415.

[116] G. Karypis, "Cluto - a clustering toolkit," University of Minnesota, Department of Computer Science, Tech. Rep. 02-017, 2002.

[117] D. Greene and P. Cunningham, "Practical solutions to the problem of diagonal dominance in kernel document clustering," in *Proc. 23rd International Conference on Machine learning (ICML'06)*. ACM Press, 2006, pp. 377–384.

[118] Y. Zhao and G. Karypis, "Empirical and theoretical comparisons of selected criterion functions for document clustering," *Machine Learning*, vol. 55, no. 3, pp. 311–331, 2004.

[119] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[120] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[121] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.

[122] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.