Alexandria University

Graduation Project

A Proposed Approach for Arabic Semantic Similarity Applied to News Reommender System

Supervisor:

Prof. Dr. Nagwa El-Makky
Dr. Noha A. Yousri

Members:

Ahmed Mohamed Mohsen
Ahmed Mahmoud El-bagoury
Ahmed Hesham Emara
Samer Samy Meggaly
Mohamed Gaber

A thesis submitted in fulfilment of the requirements for the degree of Bachelor

Computer & System Engineering Department

June 2012

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

ALEXANDRIA UNIVERSITY

Abstract

Faculty of Engineering
Computer & System Engineering Department

Bachelor of Engineering

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

Abstract ii

Acknowledgements iii

List of Figures viii

List of Tables ix

Abbreviations x

Physical Constants xi

Symbols xii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Goals and Scoop . . . 2
- 1.3 Structure of the document . . . 2

2 Literature 3
- 2.1 Clustering . . . 3
- 2.2 Semantic Similarity . . . 5
- 2.3 Recommendation . . . 6
  - 2.3.1 Definition of a Recommender System . . . 6
  - 2.3.2 Basics to a Recommender System . . . 7
  - 2.3.3 Challenges of Recommender Systems . . . 7
    - 2.3.3.1 Quality of a recommender system . . . 7
    - 2.3.3.2 Sparsity . . . 8
    - 2.3.3.3 Synonymy . . . 8
    - 2.3.3.4 First Rater Problem . . . 8
  - 2.3.4 Memory Based Recommender Systems . . . 8
    - 2.3.4.1 UserBased Recommendation . . . 8
    - 2.3.4.2 ItemBased Recommendation . . . 9
  - 2.3.5 Comparison between Userbased and Itembased recommender systems . . . 10
  - 2.3.6 ContentBased Recommender Systems . . . 10
  - 2.3.7 Testing the Accuracy of a Recommender System . . . 10

2.3.7.1 Accuracy . . . 11
2.3.7.2 Mean Absolute Error . . . 11

3 Document Similarity **12**
3.1 Lexical Similarity . . . 12
3.1.1 Document Represenatation . . . 12
3.1.2 Similarity Measures . . . 13
3.1.2.1 Eculidean Distance . . . 13
3.1.2.2 Cosine Similarity . . . 13
3.2 Semantic Similarity . . . 14
3.2.1 Corpus Based . . . 16
3.2.1.1 LSA for Semantic Similarity . . . 16
3.2.2 Knowledge Based . . . 18
3.2.2.1 Wu & Palmer . . . 19
3.2.2.2 AWN . . . 20
3.2.2.3 Wikipedia . . . 24
3.2.3 Term Weigthing using Specificity . . . 26
3.2.3.1 Latent Semantic Analysis: . . . 26
3.2.3.2 Inverse Document Frequency: . . . 27
3.2.4 Disambiguation . . . 28
3.2.4.1 JIGSAW Algorithm . . . 28

4 Document Clustering **30**
4.1 Dynamic modeling and distance relatedness concepts . . . 31
4.1.1 Dynamic vs. static models . . . 31
4.1.2 Distance-based density . . . 32
4.2 Proposed distance relatedness dynamic model . . . 32
4.2.1 Distance consistency definitions . . . 34
4.3 Algorithms phases . . . 34
4.4 Implementation Details . . . 35
4.4.1 Disjoint set . . . 35
4.4.1.1 Linked-List Implementation . . . 36
4.4.1.2 Disjoint-set Implementation: Forests . . . 36

5 Recommender Systems **38**
5.1 Introduction . . . 38
5.2 Recommendation Problem . . . 38
5.3 Content-Based Methods . . . 40
5.3.1 Limitation . . . 43
5.3.1.1 Limited Content Analysis . . . 43
5.3.1.2 Over-Specialization . . . 43
5.3.1.3 New User Problem . . . 44
5.4 Collaborative Methods . . . 44
5.4.1 Collaborative recommendations algorithms . . . 45
5.4.2 Computing similarity between users . . . 46
5.4.2.1 Correlation based similarity . . . 46
5.4.2.2 Cosine based similarity . . . 46

5.4.3 Cluster models . . . 47
5.4.4 Bayesian networks . . . 47
5.4.5 Default Voting . . . 47
5.4.6 Limitation . . . 48
5.4.6.1 New user problem . . . 48
5.4.6.2 New item problem . . . 48
5.4.6.3 Sparsity . . . 48
5.5 Hybrid Methods . . . 49
5.5.1 Combining separate recommenders . . . 49
5.5.2 Adding content-based characteristics to collaborative models . . . 50
5.5.3 Adding collaborative characteristics to content-based models . . . 50
5.5.4 Developing a single unifying recommendation model . . . 50
5.6 Demographic-based Methods . . . 50
5.7 Collaboration via content . . . 51
5.8 Building user Profile . . . 51
5.9 User's Feedback . . . 53
5.10 Collaboration via content . . . 53
5.10.1 Explicit feedback . . . 53
5.10.2 Implicit feedback . . . 53
5.11 Using LSA in Recommendation . . . 53
5.11.1 Advantage of using LSA . . . 53

6 Tools **55**
6.1 AWN . . . 55
6.2 Crawlers . . . 56
6.2.1 Basic concept . . . 56
6.3 Corpus . . . 58
6.4 LSA Tools . . . 58
6.4.1 Python . . . 58
6.4.1.1 Description . . . 58
6.4.1.2 Usage . . . 58
6.4.2 DIVISI2 . . . 59
6.4.2.1 Description . . . 59
6.4.2.2 Usage . . . 59
6.4.3 PyTables . . . 59
6.4.3.1 Description . . . 59
6.4.3.2 Usage . . . 59
6.4.4 Numpy . . . 59
6.4.4.1 Description . . . 59
6.4.4.2 Usage . . . 60

A Appendix Title Here **61**
A.1 Crawlers XML Configuration File . . . 61
A.2 JIGSAW Disambiguation . . . 64

Bibliography **66**

List of Figures

2.1 Clustering techniques 3

3.1 Angle Between Documents 13

4.1 Distances between patterns reflect the density structure. 33
4.2 Effect of distance inconsistency on merging of patterns 34
4.3 Two sets 36
4.4 Union of two sets 36
4.5 Path compression 37

List of Tables

2.1 User Item Matrix 9
2.2 Similarity matrix based on values from Table 2.1 9

5.1 A fragment of a rating matrix for a movie recommender system 39
5.2 The four essential requirements 52

Abbreviations

LAH **L**ist **A**bbreviations **H**ere

Physical Constants

Speed of Light $c = 2.997\ 924\ 58 \times 10^8\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	$rads^{-1}$

For/Dedicated to/To my...

Chapter 1

Introduction

A general overview of the documentation is provided in this chapter, focusing on the definition of the problems that motivated the work. Section 1.1 presents the motivation which made us interested in this work, and shows the limitations of the previous work.
Section 1.2 states the goals of our work to be achieved. Section 1.3 describes the structure of this document.

1.1 Motivation

With the tremendous increase of the on-line news streams, the need to aggregate related news has also increased, also the need to filter duplicate news. Here arouses the role of recommendation systems which help the readers surf the news that are likely to be of interest. Systems recommend items of interest to users based on preferences they have expressed either explicitly or implicitly. Such systems have an obvious appeal in situations where the amount of on-line news available to users greatly exceeds the users ability to survey it.
On contrast to the existence of many systems that support the English language, only a few can be found for Arabic language in spite of its importance. Arabic language consists of 28 letters and is used by more than 330 million Arabic speakers that are spread over 22 countries (Ghosn, 2003; Censure of the Internet in the Arab countries, 2006). The performance of information retrieval in Arabic language is very problematic which lead to the arousal of many challenges in developing text analysis and recommendation systems for Arabic documents. The complex and rich nature of the Arabic language can be observed in the morphological and structural changes in the language like polysemy, irregular and inflected derived forms, various spelling of certain words, various writing of certain characters combination, short (diacritics) and long vowels. In addition, most of the Arabic words contain affixes. The language is written from right to left. Moreover, the majority of words have a tri-letter root. The rest have either a quad-letter root,

penta-letter root or hexa-letter root.
Similarity between documents is one of the issues in information retrieval and a major issue in recommendation systems. Almost all of the proposed systems for Arabic are based on Lexical Similarity which is weakened by the complex nature of the language. Another approach that provides promising results is similarity based on the Semantics of the context. Semantics of the context helps capture the essence of the document. Hence, Semantic Similarity provides a better measure of affinity between Arabic documents.

1.2 Goals and Scoop

In this work we focus on evaluating semantic similarity techniques on Arabic text. We chose news articles as a case study because they are written mainly in a formal non-colloquial Arabic and to avoid the variation of Arabic dialects. We considered the following points as main goals

- Building semantic similarity module using two different approaches: knowledge based and corpus based.
- Using the semantic similarity as an affinity metric to cluster documents and users' profiles.
- Recommend news to users based on her feed back and predict her taste.

1.3 Structure of the document

Chapter 2

Chapter 2

Literature

2.1 Clustering

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups *clusters*, such that items within a cluster are very similar and items in different clusters are very different. In this work we used clustering to group similar documents together and to determine the user neighborhood which affects the scalability of large scale recommender system.

Different approaches to clustering data can be described with the help of the hierarchy shown in Figure 2.1

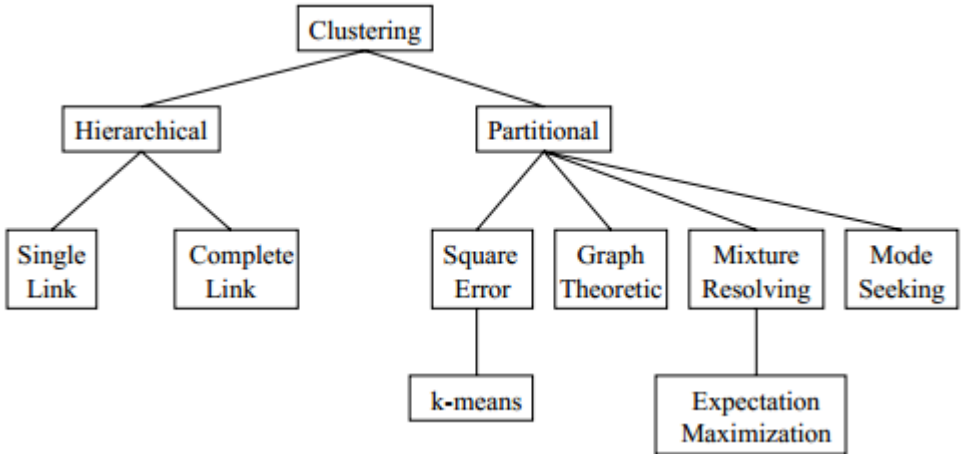


FIGURE 2.1: Clustering techniques

- Partitional : Given a database of objects, a partitional clustering algorithm constructs partitions of the data, where each cluster optimizes a clustering criterion, such as the

minimization of the sum of squared distance from the mean within each cluster. One of the issues with such algorithms is their high complexity, as some of them exhaustively enumerate all possible groupings and try to find the global optimum. Even for a small number of objects, the number of partitions is huge. Thats why, common solutions start with an initial, usually random, partition and proceed with its refinement. A better practice would be to run the partitional algorithm for different sets of initial points (considered as representatives) and investigate whether all solutions lead to the same final partition. Partitional Clustering algorithms try to locally improve a certain criterion. First, they compute the values of the similarity or distance, they order the results, and pick the one that optimizes the criterion.Hence, the majority of them could be considered as greedy-like algorithms.

- Hierarchical algorithms create a hierarchical decomposition of the objects. They are either agglomerative (bottom-up) or divisive (top-down):
  - Agglomerative algorithms start with each object being a separate cluster itself, and successively merge groups according to a distance measure. The clustering may stop when all objects are in a single group or at any other point the user wants. These methods generally follow a greedy-like bottom-up merging.
  - Divisive algorithms follow the opposite strategy. They start with one group of all objects and successively split groups into smaller ones, until each object falls in one cluster, or as desired. Divisive approaches divide the data objects in disjoint groups at every step, and follow the same pattern until all objects fall into a separate cluster. This is similar to the approach followed by divide-and-conquer algorithms. Partitional and hierarchical methods can be integrated. This would mean that a result given by a hierarchical method can be improved via a partitional step, which refines the result via iterative relocation of points.

Apart from the two main categories of partitional and hierarchical clustering algorithms, many other methods have emerged in cluster analysis, and are mainly focused on specific problems or specific data sets available. These methods include:

- Density-Based Clustering : These algorithms group objects according to specific density objective functions. Density is usually defined as the number of objects in a particular neighborhood of a data objects. In these approaches a given cluster continues growing as long as the number of objects in the neighborhood exceeds some parameter. This is considered to be different from the idea in partitional algorithms that use iterative relocation of points given a certain number of clusters.
- Grid-Based Clustering : The main focus of these algorithms is spatial data, i.e., data that model the geometric structure of objects in space, their relationships, properties and

operations. The objective of these algorithms is to quantize the data set into a number of cells and then work with objects belonging to these cells. They do not relocate points but rather build several hierarchical levels of groups of objects. In this sense, they are closer to hierarchical algorithms but the merging of grids, and consequently clusters, does not depend on a distance measure but it is decided by a predefined parameter.

- Model-Based Clustering : These algorithms find good approximations of model parameters that best fit the data. They can be either partitional or hierarchical, depending on the structure or model they hypothesize about the data set and the way they refine this model to identify partitionings. They are closer to density-based algorithms, in that they grow particular clusters so that the preconceived model is improved. However, they sometimes start with a fixed number of clusters and they do not use the same concept of density.
- Categorical Data Clustering : These algorithms are specifically developed for data where Euclidean, or other numerical-oriented, distance measures cannot be applied. In the literature, we find approaches close to both partitional and hierarchical methods.

2.2 Semantic Similarity

The problem of formalizing and quantifying the intuitive notion of similarity has a long history in philosophy, psychology, and artificial intelligence, and many different perspectives have been suggested. Recent research on the topic in computational linguistics has emphasized the perspective of semantic relatedness of two lexemes in a lexical resource, or its inverse, semantic distance. Its important to note that semantic relatedness is a more general concept than similarity; similar entities are usually assumed to be related by virtue of their likeness (bank - trust company), but dissimilar entities may also be semantically related by lexical relationships such as metonymy (car - wheel) and antonymy (hot - cold), or just by any kind of functional relationship or frequent association (pencil - paper, penguin - Antarctica).

Measures of text similarity have been used for a long time in applications in natural language processing and related areas. Text similarity has been also used for relevance feedback and text classification (Rocchio, 1971), word sense disambiguation (Lesk, 1986), and more recently for extractive summarization (Salton et al., 1997b), and methods for automatic evaluation of machine translation (Papineni et al., 2002) or text summarization (Lin and Hovy, 2003).
The typical approach to finding the similarity between two text segments is to use a simple lexical matching method, and produce a similarity score based on the number of lexical units that occur in both input segments.
Improvements to this simple method have considered stemming, stop-word removal, part-of-speech tagging, longest subsequence matching, as well as various weighting and normalization

factors (Salton et al., 1997a).
While successful to a certain degree, these lexical matching similarity methods fail to identify the semantic similarity of texts. For instance, there is an obvious similarity between the text segments I own a dog and I have an animal, but most of the current text similarity metrics will fail in identifying any kind of connection between these texts.

The only exception to this trend is perhaps the latent semantic analysis (LSA) method (Landauer et al., 1998), which represents an improvement over earlier attempts to use measures of semantic similarity for information retrieval (Voorhees, 1993), (Xu and Croft, 1996). LSA aims to find similar terms in large text collections, and measure similarity between texts by including these additional related words.
However, to date LSA has not been used on a large scale, due to the complexity and computational cost associated with the algorithm, and perhaps also due to the black-box effect that does not allow for any deep insights into why some terms are selected as similar during the singular value decomposition process.

On the other hand, another group of approaches that gaines wide approval is the Knowledge-based group. Knowledge-based can be explained as a measure of semantic similarity between words, and an indication of the word specificity.

2.3 Recommendation

As the World Wide Web grows, recommender systems are becoming an increasingly important aspect to all e-commerce portals. Popular purchasing sites such as Amazon (www.amazon.com) and eBay (www.ebay.com) represent some of the businesses that have integrated recommendations into their shopping experience. These systems help to overcome excess information overload by providing customers with recommendations or suggestions based on their likes and dislikes relative to other customers. Recommendations have become an integral aspect of these e-commerce platforms and have shown to personalize the shopping experience.(Schafer, Konstan and Riedi 1999)

2.3.1 Definition of a Recommender System

Recommender systems are a type of information filtering system that gives advice on products, information, or services that a user may be interested in. They assist users ith the decision

making process when choosing items with multiple alternatives(Werthner, Hansen and Ricci 2007). Recommender systems are popular due to their e-commerce application purposes. Within the e-commerce world, recommendations provide aid to customers and help them find what they may be looking for, thus increasing business. In addition, it can be used as a tool to predict user's behavior, but should not be used to select recommendations on their behalf. There are two basic entities that appear in any recommender system are the user and the item of interest. The user can be a customer in an e-commerce platform or an avid book reader looking for a recommendation for the next book they should read. The users provide their ratings on items and are used to aid other users with their recommendations. The item is the second piece of a recommender system. Users give items ratings and the algorithm outputs recommended items based on new user queries.

2.3.2 Basics to a Recommender System

All recommender systems, whether they are in the e-commerce world or any other application, all have three things in common: they all take in inputs; they all have a goal; and they all produce and output (Vozalis and Margaritis 2003). The input in a recommender system can depend on the type of recommender system algorithm. Generally, inputs belong to one of the following categories: 1.Votes/Ratings: Ratings are based on the opinions of users on specific items that they have reviewed. 2.Demographic data: Information such as age, gender, education, and location of the user are factors that a recommender system can utilize. 3.Content data: textual analysis of documents related to the item that has been rated by the user. This information will also assist in the recommender system. The goal of a recommender system is to predict what the user will rate an item or generate suggestions about these items. The goal is generated based on the input of previous users within a user-item matrix. This matrix is used to correlate users/items to the current user/item and generate a recommendation. Finally, the output of a recommender system can either be a prediction or the expected rating score the active user will give the current item. A recommendation is a single or list of items that have the highest prediction.

2.3.3 Challenges of Recommender Systems

2.3.3.1 Quality of a recommender system

Can the recommender system be trusted to produce accurate recommendations? The recommender system must eliminate recommendations produced by the system that the system believes the user will like but in actually the user does not like. These are also known as false positives. (Vozalis and Margaritis 2003)

2.3.3.2 Sparsity

Since users may not rate some items, the user-item matrix may have many missing ratings and be very sparse. Therefore, finding correlations between users and items becomes quite difficult and can lead to weak recommendations. (Melleville,Mooney and Nagarajan 2002)

2.3.3.3 Synonymy

Recommender systems are usually not able to discover associations between similar items that may just have different names.

2.3.3.4 First Rater Problem

An item cannot be recommended unless it has been rated before. The problem usually occurs when new items are added to the system or when there are items that are rarely viewed and therefore may not have been rated. (Melleville, Mooney and Nagarajan 2002)

2.3.4 Memory Based Recommender Systems

Recommender systems that utilize memory-based algorithms and utilize a user-item matrix to generate a prediction fall under the memory based recommender systems category. The majority of such systems belong to Collaborative Filtering (CF) systems. There are three steps into processing a recommendation based on a CF system: 1.Representation 2.Neighborhood Formation 3.Recommendation Generation

2.3.4.1 UserBased Recommendation

A user-based recommendation system involves seeing how well users correlate to each other and using this information to derive predictions. A running example will demonstrate how a user-based recommendation can easily generate a prediction using a mathematical algorithm. Representation Since there are two aspects to a recommender system as noted above. The users and the items can be placed as a collection of numerical ratings into a user-item matrix.

As seen in Table 1, a user-item matrix allows for the creation of a neighborhood that can be used to create similarity between users whom we wish to generate recommendations for. The user-item matrix is usually sparse since most users do not rate every item or may not have used the specific item. Sparsity can be an issue that can lead to weak recommendations. Two common processes

Table 2.1: User Item Matrix

	Item1	Item2	Item3
John	2	3	5
Bob	3	4	3
Lucy	5	2	Did not rate

that can reduce the sparsity issue and improve the satisfaction of the recommender system are: 1.Default Voting: Setting an appropriate rating for all items that have not been rated. The cost of applying a default rating is low (Vozalis and Margaritis 2003). - Table 1: Providing Lucy with a default value of 3 for Item 3. 2.Pre-processing using averages: Scan through missing items and either averaging users votes or averaging the item votes and placing the resulting average as the rating value for the missing user-item matrix entry (Vozalis and Margaritis 2003) - Table 1: Lucys other items provide an average of 3.5. Therefore, we can pre-process her rating for Item 3. Neighbourhood Formation Generating a neighborhood involved calculating the similarity between the given users within the user-item matrix. Similarity will be used to generate a recommendation for a specific user. The algorithm follows these steps: 1.Compare the similarity between all users with the active user. 2.Select n users that have the highest similarity to build a neighborhood 3.Compute the prediction based on this similarity matrix. Within the user-based recommendation system, similarity between two users is calculated using the Pearsons correlation coefficient. Pearsons correlation (also called the Pearson's product moment correlation after Karl Pearsons) has become a standard way of calculating correlation.

Table 2.2: Similarity matrix based on values from Table 2.1

	John	Bob	Lucy
John	1	-0.1887	-0.5146
Bob		1	-0.9486
Lucy			1

Generate Prediction The prediction is a numerical value that represents a predicted opinion of the active user about a specific item. The prediction for a user-based collaborative filtering algorithm needs both the user-item matrix (Table 1) and the similarity matrix (Table 2).

2.3.4.2 ItemBased Recommendation

This form of collaborative filtering is computed based on item relations and not on user relations. An item-based collaborative filtering algorithm looks at the set of items that an active user has rated and computes how similar the set of items are to the target item. Using this data, the item-based algorithm then uses the similarity and calculates the prediction based on weighted averages of the active users ratings on those similar items. (Sarwar, et al. 2001) The prediction algorithm

for item-based collaborative filtering follows the same procedure as the user-based collaborative filtering. Item-based collaborative filtering follows the same three procedures: representation, neighborhood formation, and prediction generation. Within representation, item-based CF uses the user-item matrix seen in Table 1. The neighborhood formation creates a similarity matrix. This similarity matrix uses the Pearson Correlation coefficient, however, determines the correlation between two items

2.3.5 Comparison between Userbased and Itembased recommender systems

There are two major factors that come into play when talking about how user-based recommender systems compare to item-based recommender systems - the quality of the results and the performance results. In an experiment performed by the GroupLens Research group, it was conclusive that based on building a user-based and an item-based recommender system, the item-based recommender system provided better quality of results with a lower MAE (Mean Accuracy Error) than the user-based recommender system(Sarwar, et al. 2001). In terms of performance, the GroupLens Research group has also concluded that an item based recommender system also has better performance over a user-based system since the item neighborhood is fairly static and can potentially be pre-computed, which results in higher online performance(Sarwar, et al. 2001)

2.3.6 ContentBased Recommender Systems

Content-based recommendation systems or content-based filtering is based on textual information such as documents. These items are typically described with keywords and weights. Using nearest neighbor functions or clustering methods can allow this recommendation system to analyze these keywords and document content and use it as a basis to recommend a suitable item. This will also be based on the items characteristics. The different techniques used in this method of information filtering follow two approaches: Heuristic-based and model-based methods. Heuristic-based methods include KNN algorithms and clustering methods while model-based methods include Bayesian filtering, artificial neural networks, and clustering. Content-based filtering systems are limited by their content. If they have limited keywords or overspecialization problems, they produce weak recommendations. (Wei, Jinghu and Shaohung 2007)

2.3.7 Testing the Accuracy of a Recommender System

There are multiple ways to evaluate the predictive quality of recommender systems. We usually evaluate a recommender system based on accuracy.

2.3.7.1 Accuracy

To measure a recommender systems accuracy, we usually measure how effective the systems predictions are to actual results. A common way of measuring accuracy is by using a statistical accuracy metric called Mean Accuracy Error (MAE).

2.3.7.2 Mean Absolute Error

MAE is measured for items that the user has expressed opinions and given a rating to using the following equation (Vozalis and Margaritis 2003). The recommender system has to have generated its list of predictions and the actual ratings have to be provided as well. A lower MAE corresponds to a more accurate prediction. MAEi=j=1ni—arij-rij—ni Where ni is the number of items that the user has expressed ratings on, rij are the predictions generated for the chosen items and arij are the actual ratings provided by the user for those chosen items. In terms of grades, which will be the case in this study, a MAE of 5 will mean that the predicted grades are on average different from the actual grades by about 5 marks. We can use the MAE to determine if specific sets of data to determine if the model improves as we enter more ratings for that specific user. With these tests and assessments, we can determine if our recommender system is performing well or if adjustments need to be made in order to improve accuracy.

Chapter 3

Document Similarity

3.1 Lexical Similarity

Similarity Measures for Text Document Clustering Anna Huang Department of Computer Science The University of Waikato, Hamilton, New Zealand lh92@waikato.ac.nz

In linguistics, lexical similarity is a measure of the degree to which the word sets of two given languages are similar. A lexical similarity of 1 (or 100%) would mean a total overlap between vocabularies, whereas 0 means there are no common words. (wiki)

3.1.1 Document Represenatation

There are several ways to model a text document. For example, it can be represented as a bag of words, where words are assumed to appear independently and the order is immaterial. The bag of word model is widely used in information retrieval and text mining [21]. Words are counted in the bag, which differs from the mathematical definition of set. Each word corresponds to a dimension in the resulting data space and each document then becomes a vector consisting of non-negative values on each dimension. Here we use the frequency of each term as its weight, which means terms that appear more frequently are more important and descriptive for the document. Let D $= \{d_1, ..., d_n\}$ be a set of documents and T $= \{t_1, ..., t_m\}$ the set of distinct terms occurring in D. We discuss more precisely what we mean by "terms" below: for the moment just assume they are words. A document is then represented as a m-dimensional vector t_d.
Let tf (d, t) denote the frequency of term $t \epsilon T$ in document $d \epsilon D$. Then the vector representation of a document d is

$$\vec{t_a} = (tf(d, t_1, ..., tf(d, t_m)) \tag{3.1}$$

Although more frequent words are assumed to be more important as mentioned above, this is not usually the case in practice. For example, words like a and the are probably the most frequent words that appear in English text, but neither are descriptive nor important for the documents subject. In fact, more complicated strategies such as the tfidf weighting scheme as described

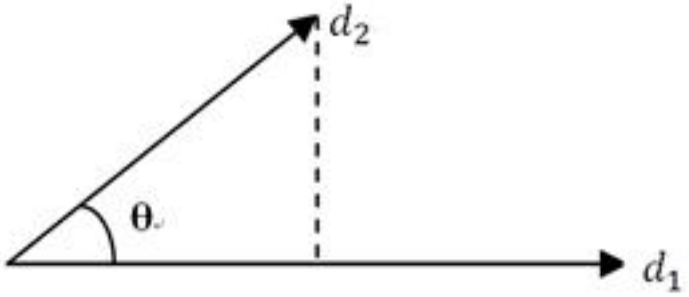


FIGURE 3.1: Angle Between Documents

below is normally used instead. With documents presented as vectors, we measure the degree of similarity of two documents as the correlation between their corresponding vectors, which can be further quantified as the cosine of the angle between the two vectors. Figure 1 shows the angle in two-dimensional space but in practice the document space usually has tens and thousands of dimensions. Some useful properties of the cosine measure are discussed in Section 3.3.

3.1.2 Similarity Measures

3.1.2.1 Eculidean Distance

Euclidean distance is a standard metric for geometrical problems. It is the ordinary distance between two points and can be easily measured with a ruler in two- or three-dimensional space. Euclidean distance is widely used in clustering problems, including clustering text. It satisfies all the above four conditions and therefore is a true metric. It is also the default distance measure used with the K-means algorithm.
Measuring distance between text documents, given two documents $d_a and d_b$ represented by their term vectors $\overrightarrow{t_a}\, and\, \overrightarrow{t_b}$ respectively, the Euclidean distance of the two documents is defined as where the term set is T = t1 , . . . , tm . As mentioned previously, we use the tf idf value as term weights, that is crucial for cluster analysis, especially for a particular type $w_{t,a} = tfidf(d_a, t)$.

3.1.2.2 Cosine Similarity

documents are represented as term vectors, the similarity of two documents corresponds to the correlation between the vectors. This is quantified as the cosine of the angle between vectors,

that is, the so-called cosine similarity. Cosine similarity is one of the most popular similarity measure applied to text documents, such as in numerous information retrieval applications [21] and clustering too [9]. Given two documents $\overrightarrow{t_a}$ *and* $\overrightarrow{t_b}$, their cosine similarity is

$$SIM_c(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a}.\vec{t_b}}{|\vec{t_a}| \times |\vec{t_b}|} \tag{3.2}$$

where $\overrightarrow{t_a}$ *and* $\overrightarrow{t_b}$ are m-dimensional vectors over the term set T = {t1 , ... , tm }. Each dimension represents a term with its weight in the document, which is non-negative. As a result, the cosine similarity is non-negative and bounded between [0,1]. An important property of the cosine similarity is its independence of document length. For example, combining two identical copies of a document d to get a new pseudo document d , the cosine similarity between d and d is 1, which means that these two documents are regarded to be identical. Meanwhile, given another document l, d and d will

3.2 Semantic Similarity

Corpus-based and Knowledge-based Measures of Text Semantic Similarity Rada Mihalcea and Courtney Corley Carlo Strapparava Department of Computer Science Istituto per la Ricerca Scientifica e Tecnologica University of North Texas ITC irst rada,corley@cs.unt.edu strappa@itc.it

Measures of semantic similarity have been traditionally defined between words or concepts, and much less between text segments consisting of two or more words. The emphasis on word-to-word similarity metrics is probably due to the availability of resources that specifically encode relations between words or concepts (e.g. WordNet), and the various testbeds that allow for their evaluation (e.g. TOEFL or SAT analogy/synonymy tests). Moreover, the derivation of a text-to-text measure of similarity starting with a word based semantic similarity metric may not be straightforward,and consequently most of the work in this area has considered mainly applications of the traditional vectorial model,occasionally extended to n-gram language models. Given two input text segments, we want to automatically derive a score that indicates their similarity at semantic level,thus going beyond the simple lexical matching methods traditionally used for this task. Although we acknowledge the fact that a comprehensive metric of text semantic similarity should also take into account the structure of the text, we take a first rough cut at this problem and attempt to model the semantic similarity of texts as a function of the semantic similarity of the component words. We do this by combining metrics of word-to-word similarity and word specificity into a formula that is a potentially good indicator of the semantic similarity of the two input texts.

The following section provides details on eight different corpus-based and knowledge-based measures of word semantic similarity. In addition to the similarity of words, we also take into account

the specificity of words, so that we can give a higher weight to a semantic matching identified between two specific words (e.g. collie and sheepdog), and give less importance to the similarity measured between generic concepts (e.g. get and become). While the specificity of words is already measured to some extent by their depth in the semantic hierarchy, we are reinforcing this factor with a corpus-based measure of word specificity, based on distributional information learned from large corpora.The specificity of a word is determined using the inverse document frequency (idf) introduced by Sparck-Jones (1972), defined as the total number of documents in the corpus divided by the total number of documents including that word. The idf measure was selected based on previous work that theoretically proved the effectiveness of this weighting approach (Papineni 2001). In the experiments reported here, document frequency counts are derived starting with the British National Corpus a 100 million words corpus of modern English including both spoken and written genres.

Given a metric for word-to-word similarity and a measure of word specificity, we define the semantic similarity of two text segments T1 and T2 using a metric that combines the semantic similarities of each text segment in turn with respect to the other text segment. First, for each word w in the segment T1 we try to identify the word in the segment T2 that has the highest semantic similarity (maxSim(w, T2)), according to one of the word-to-word similarity measures described in the following section. Next, the same process is applied to determine the most similar word in T1 starting with words in T2 . The word similarities are then weighted with the corresponding word specificity, summed up, and normalized with the length of each text segment. Finally the resulting similarity scores are combined using a simple average. Note that only open-class words and cardinals can participate in this semantic matching process. As done in previous work on text similarity using vector-based models, all function words are discarded. The similarity between the input text segments $T_1 and T_2$ is therefore determined using the following scoring function:

$$SIM(T_1, T_2) = \frac{1}{2}\left(\frac{\sum_{w\in\{T_1\}}(maxSim(w, T_2 * idf(w)))}{\sum_{w\in\{T_1\}} idf(w)} + \frac{\sum_{w\in\{T_2\}}(maxSim(w, T_1 * idf(w)))}{\sum_{w\in\{T_2\}} idf(w)}\right) \tag{3.3}$$

This similarity score has a value between 0 and 1, with a score of 1 indicating identical text segments, and a score of 0 indicating no semantic overlap between the two segments.Note that the maximum similarity is sought only within classes of words with the same part-of-speech. The rea- son behind this decision is that most of the word-to-word knowledge-based measures cannot be applied across parts-of-speech, and consequently, for the purpose of consistency,we imposed the same word-class restriction to all the word-to-word similarity measures. This means that, for instance, the most similar word to the noun flower within the text There are many green plants next to the house will be sought among the nouns plant and house, and will ignore the words with a different part-of-speech (be, green, next). Moreover, for those parts-of-speech for which a

word-to-word semantic similarity cannot be measured (e.g. some knowledge-based measures are not defined among adjectives or adverbs), we use instead a lexical match measure, which assigns a maxSim of 1 for identical occurrences of a word in the two text segments.

3.2.1 Corpus Based

This LaTeX Thesis Template is originally based and created around a LaTeX style file created by Steve R. Gunn from the University of Southampton (UK), department of Electronics and Computer Science. You can find his original thesis style file at his site, here:
http://www.ecs.soton.ac.uk/~srg/softwaretools/document/templates/

My thesis originally used the '`ecsthesis.cls`' from his list of styles. However, I knew LaTeX could still format better. To get the look I wanted, I modified his style and also created a skeleton framework and folder structure to place the thesis files in.

This Thesis Template consists of that modified style, the framework and the folder structure. All the work that has gone into the preparation and groundwork means that all you have to bother about is the writing.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change to bring the template in line with your institution's recommendations.

3.2.1.1 LSA for Semantic Similarity

LSA is a fully automatic mathematical/statistical technique for extracting and inferring relations of expected contextual usage of words in passages of discourse. It is not a traditional natural language processing or artificial intelligence program; it uses no humanly constructed dictionaries, knowledge bases, semantic networks, grammars, syntactic parsers, or morphologies, or the like, and takes as its input only raw text parsed into words defined as unique character strings and separated into meaningful passages or samples such as sentences or paragraphs. Another way to think of this is that LSA represents the meaning of a word as a kind of average of the meaning of all the passages in which it appears, and the meaning of a passage as a kind of average of the meaning of all the words it contains. LSA's ability to simultaneously derive representations of these two interrelated kinds of meaning depends on an aspect of its mathematical machinery. How LSA works The first step is to represent the text as a matrix (the term-by-document matrix T)in which each row stands for a unique word and each column stands for a text passage or other context. Each cell contains the frequency with which the word of its row appears in the passage denoted by its column. Next, the cell entries are subjected to a preliminary transformation,

whose details we will describe later, in which each cell frequency is weighted by a function that expresses both the word's importance in the particular passage and the degree to which the word type carries information in the domain of discourse in general. Next, LSA applies singular value decomposition (SVD) to the matrix. This is a form of factor analysis, or more properly the mathematical generalization of which factor analysis is a special case. In SVD, a rectangular matrix is decomposed into the product of three other matrices. One component matrix (U) describes the original row entities as vectors of derived orthogonal factor values, another (V) describes the original column entities in the same way, and the third is a diagonal matrix (SIGMA) containing scaling values such that when the three components are matrix-multiplied, the original matrix (T) is reconstructed. There is a mathematical proof that any matrix can be so decomposed perfectly, using no more factors than the smallest dimension of the original matrix. When fewer than the necessary number of factors are used, the reconstructed matrix is a least-squares best fit.

One can reduce the dimensionality of the solution simply by deleting coefficients in the diagonal matrix (SIGMA), ordinarily starting with the smallest. (In practice, for computational reasons, for very large corpora only a limited number of dimensions, currently a few thousands can be constructed). The SVD equations can be derived as follows:

SVD Equation

Stop-listing and stemming These are very rarely used. In keeping with the underlying theory and model, neither stemming nor stop-listing is appropriate or usually effective. As in natural language, the meaning of passages cannot be accurately reconstructed or understood without all of its words. However, when LSA is used to compare word strings shorter than normal text paragraphs, e.g. short sentences, zero weighting of function words is often pragmatically useful. Similarity in the reduced space Since both passages (documents) and terms (words) are represented as vectors, it is straight forward to compute the similarity between passage-passage, term-term, and term-passage. In addition, terms and/or passages can be combined to create new vectors in the space. The process by which new vectors can be added to an existing LSA space is called folding-in. The cosine distance between vectors is used as the measure of their similarity for many applications because of its relation to the dot-product criterion and has been found effective in practice, although other metrics, such as Euclidean distance or City-Block (Manhattan) distance are sometimes used. Shortfalls, objections, evidence and arguments otential limitations should be noted: 1. It is likely, of course, that LSA's bag of words method, that ignores all syntactical, logical and nonlinguistic pragmatic entailments, sometimes misses meanings or gets it scrambled. 2. LSA became practical only when computational power and algorithm efficiency improved sufficiently to support SVD of thousands of words-by-thousands of

contexts matrices; it is still impossible to perform SVD on the hundreds of thousands by tens of millions matrices that would be needed to truly represent the sum of an adult's language exposure.

Some commentators have also argued that LSA must be fundamentally wrong as theory because is not grounded in perception and intention. The strength of this objection is considerably reduced by the observation that language must veridically reflect these sources or it would be nearly useless, and by the human ability to generate, use and understand words as abstract and unrelated to perception as the word abstract itself, and by LSAs varied successes.

Indeed, one should not think of LSA as a fixed mechanism or its representations as fixed quantities, but rather, as evolving approximations.

3.2.2 Knowledge Based

Evaluating semantic relatedness using network representations is a problem with a long history in arti cial intelligence and psychology, dating back to the spreading activation approach of Quillian (1968) and Collins and Loftus (1975). Semantic similarity represents a special case of semantic relatedness: for example, cars and gasoline would seem to be more closely related than, say, cars and bicycles, but the latter pair are certainly more similar.
Rada et al. (Rada, Mili, Bicknell, Blettner, 1989) suggest that the assessment of similarity in semantic networks can in fact be thought of as involving just taxonomic (is-a) links, to the exclusion of other link types; that view will also be taken here, although admittedly links such as part-of can also be viewed as attributes that contribute to similarity (cf.Richardson, Smeaton, Murphy, 1994; Sussna, 1993).

Although many measures of similarity are de ned in the literature, they are seldom accompanied by an independent characterization of the phenomenon they are measuring, particularly when the measure is proposed in service of a computational application (e.g.,similarity of documents in information retrieval, similarity of cases in case-based reasoning).
Rather, the worth of a similarity measure is in its utility for the given task. In the cognitive domain, similarity is treated as a property characterized by human perception and intuition,in much the same way as notions like " and ." As such, the worth of a similarity measure is in its delity to human behavior, as measured by predictions of human performance on experimental tasks. The latter view underlies the work in this article, although the results presented comprise not only direct comparison with human performance but also practical application to problems in natural language processing.
A natural, time-honored way to evaluate semantic similarity in a taxonomy is to measure the distance between the nodes corresponding to the items being compared — the shorter the path

from one node to another, the more similar they are. Given multiple paths, one takes the length of the shortest one (Lee, Kim, Lee, 1993; Rada Bicknell, 1989; Rada et al., 1989).
A widely acknowledged problem with this approach, however, is that it relies on the notion that links in the taxonomy represent uniform distances. Unfortunately, uniform link distance is di cult to de ne, much less to control. In real taxonomies, there is wide variability in the " covered by a single taxonomic link, particularly when certain sub-taxonomies (e.g., biological categories) are much denser than others. For example, in WordNet (Miller, 1990; Fellbaum, 1998), a widely used, broad-coverage semantic network for English, it is not at all di cult to nd links that cover an intuitively narrow distance (rabbit ears is-a television antenna) or an intuitively wide one (phytoplankton is-a living thing). The same kinds of examples can be found in the Collins COBUILD Dictionary (Sinclair, ed., 1987), which identi es superordinate terms for many words (e.g.,safety valve is-a valve seems much narrower than knitting machine is-a machine).
In the rst part of this article, I describe an alternative way to evaluate semantic similarity in a taxonomy, based on the notion of information content. Like the edge-counting method, it is conceptually quite simple. However, it is not sensitive to the problem of varying link distances. In addition, by combining a taxonomic structure with empirical probability estimates, it provides a way of adapting a static knowledge structure to mul- tiple contexts. Section 2 sets up the probabilistic framework and de nes the measure of semantic similarity in information-theoretic terms, and Section 3 presents an evaluation of the similarity measure against human similarity judgments, using the simple edge-counting method as a baseline. In the second part of the article, Sections 4 and 5, I describe two applications of semantic similarity to problems of ambiguity in natural language. The rst concerns a particular case of syntactic ambiguity that involves both coordination and nominal compounds, each of which is a pernicious source of structural ambiguity in English. Consider the phrase food handling and storage procedures: does it represent a conjunction of food handling and storage procedures, or does it refer to the handling and storage of food? The second application concerns the resolution of word sense ambiguity — not for words in running text, which is a large open problem (though cf. Wilks Stevenson, 1996), but for groups of related words as are often discovered by distributional analysis of text corpora or found in dictionaries and thesauri. Finally, Section 6 discusses related work.

3.2.2.1 Wu & Palmer

If words can be defined with concepts in a hierarchical structure, it is possible to measure the meaning similarity between words with an information measure based on WordNet (Resnik,1993), or structure level information based on a thesaurus (Kurohashi and Nagao, 1992). How- ever, verb meanings are difficult to organize in a hierarchical structure.
In each conceptual domain, lexicalized concepts can be organized in a hierarchical structure. Within one conceptual domain, the similarity of two concepts is defined by how closely they are

related in the hierarchy, i.e., their structural relations.

The Wu and Palmer (Wu Palmer 1994) similarity metric measures the depth of two given concepts in the WordNet taxonomy, and the depth of the least common subsumer (LCS), and combines these figures into a similarity score:

$$SIM(T_1, T_2) = \frac{2 * depth(LCS)}{depth(concept_1) + depth(concept_2)} \tag{3.4}$$

REfer: VERB SEMANTICS AND LEXICAL SELECTION,Zhibiao Wu Martha Palmer

3.2.2.2 AWN

AWN is constructed according to the methods developed for EuroWordNet (EWN;Vossen 1998) and since applied to dozens of languages around the world. The EuroWordNet approach maximizes compatibility across wordnets and focuses on manual encoding of the most complicated and important concepts. Language-specific concepts and relations are encoded as needed or desired. This results in a so-called core wordnet for Arabic with the most important synsets, embedded in a solid semantic framework. From this core wordnet, it is possible to automatically extend the coverage with high precision. Specific concepts can be linked and translated with great accuracy because the base building blocks are manually defined and translated. The approach follows a top-down procedure. Arabic Base Concepts are defined and extended via hyponymic relations to derive a core wordnet. The set of Common Base Concepts (CBCs) from the 12 languages in EWN and BalkaNet (Tufis 2004) are encoded as synsets; other language-specific concepts are added and translated manually to the closest synset(s) in Arabic. The same step is performed for all English synsets that currently have an equivalence relation in Suggested Upper Merged Ontology (SUMO). The first layers of hyponyms are chosen on the basis of linguistic and applications based criteria; the final phase completes the target set of concepts/synsets, including specific domains and named entities. Each synset construction step is followed by a validation phase, where formal consistency is checked and the coverage is evaluated in terms of frequency of occurrence and domain distribution.

Structure and organization of AWN Because AWN is to be aligned not just to Princeton WordNet (PWN) (Fellbaum 1998) but to every wordnet aligned to PWN, the database design supports multiple languages, and the user interface will be explicitly multilingual rather than bilingual as was the one described in Black and Elkateb (2004). The database structure comprises four principal entity types, item, word, form and link.

1. Items are conceptual entities, including synsets, ontology classes and instances. Besides a unique identifier, an item has descriptive information such as a gloss. Items lexicalized in different languages are distinct.

2. A word entity is a word sense, where the citation form of the word is associated with an item via its identifier.

3. A form is a special form that is considered dictionary information (not merely an inflectional variant). The forms of Arabic words that go in this table are the root and/or the broken plural form, where applicable.

4. A link relates two items, and has a type such as "equivalence," "subsuming," etc. Links connect sense items to other sense items, e.g. a PWN synset to an AWN synset, a synset to a SUMO concept, etc.

This data model has been specified in XML as an interchange format, but is also implemented in a MySQL database hosted by one of the partners. The database will be the primary deliverable of the project, and will be distributed freely to the community.

Constructing AWN The basic criteria for selecting synsets to be covered in AWN are:

- Connectivity: AWN should be as densely connected as possible by hyperonymy / hyponymy chains, etc. Most of the synsets of AWN should correspond to English WN counterparts and the overall topology of both wordnets should be similar.

- Relevance: Frequent and salient concepts have priority. Criteria will include the frequency of lexical items (both in Arabic and English) and the frequency of Arabic roots in their respective reference corpora.

- Generality: Synsets on the highest levels of WN are preferred. These criteria suggest two ways for proceeding:

- From English to Arabic: Given an English synset, all corresponding Arabic variants (if any) will be selected.

- From Arabic to English: Given an Arabic word, all its senses have to be found, and for each of these senses the corresponding English synsets have to be selected.

Both steps have to be followed throughout the construction of AWN. All AWN synsets must be manually validated (and eventually locked, when all their variants have been found) but advantage should be taken as much as possible of the available resources for guiding the construction and validation process. Once a new Arabic verb is added to AWN, several possibilities for extension arise: extensions from verbal entries, including verbal derivatives, nominalizations, verbal nouns,

etc. We also consider the most productive forms of deriving broken plurals. This can be done using a set of lexical and morphological rules. To take full advantage of these extensions short iterations will be performed. As stated in the introduction, the starting point of AWN is the manual construction of its Base Concept (BC) set from EWN and BalkaNet's CBCs. We concentrate on the most relevant terms for obtaining about 1,000 nominal and 500 verbal synsets. The second step consists of the top-down vertical extension of BC, following Farreres 2005, Diab 2004). Some preprocessing is required for this and the next phase. We mention two tasks, preparation and extension. Preparation includes the processing of the available bilingual resources and the compilation of a set of lexical and morphological rules. From the set of available bilingual dictionaries we construct a homogeneous bilingual dictionary (HBIL) that contains for each entry information on the Arabic/English word pair, the Arabic root (added manually), POS, relative frequencies and sources supporting the pairing. The set of 17 heuristic methods used in the development of EWN will be applied to HBIL (following Farreres 2005) to derive candidate Arabic words/English synsets mappings. For each mapping the information attached includes the Arabic word and root, the English synset, POS, relative frequencies, mapping score, absolute depth in WN, number of gaps between the synset and the top of the WN hierarchy, and sources containing the pair. Arabic words in bilingual resources must be normalized and lemmatized (Diab et al. 2004, Habash and Rambow 2005) but vowels and diacritics must be maintained. Arabic roots are not vowelized. Following pre-processing, the set of scored Arabic word/English synset pairs becomes the input to the manual validation step. We proceed by chunks of related units (sets of related WN synsets, e.g. hyponymy chains and sets of related Arabic words, i.e., words having the same root) instead of individual units (synsets, senses, words). Finally, AWN will be completed by filling gaps in its structure, covering specific domains, adding terminology and named entities, etc.

Shortfalls of AWN:

- high Coupling with UI
  AWN suffers a very poor design that resulted in a high coupling between the Core Module and the Graphical User Interface (GUI) of the AWN, which made it very difficult to use the code module and the database without the GUI. However, we managed to decouple the GUI from the core modules of the AWN to allow its integration with other modules of semantic similarity. The decoupling process exerted a lot of time tracing and reverse engineering the code.

- shortage of unique terms/word i.e there is so little number of words in database
  One reasonable limitation of AWN, assuming it is an initial version, is that it doesnt contain a broad collection of Arabic words and terms. It contains only 17561 unique word, distributed among 7822 synsets.This leads to a lot of missing words that have no record in AWNs database. e.g. , ,

- running time is too damn large
  Since AWN depends on the Tree Concept (a hierarchical organization of the synsets), it has an arabic tree with nodes that represent synsets, each synset contains set of words. So every time the AWN is queried by a term, in order to retrieve its synset, the whole tree is first searched for this term. If the term was not found in the tree, we seek it from database. If the term is found in the database, then the retrieved results from the database would include the path of the term in the hierarchical tree. The final step, that is to expand the retrieved path into the tree, exhausts a substantial amount of time. The time consumption is due to the way the path expansion work. The path expansion starts at the tree root node and traverses the tree through the path retrieved from the database. If a node on the path was not found in the current tree, the node is added to the tree and all its children are fetched from the database and added to the tree. The last step consumes a lot of time because it expands all the nodes children, although only one child is needed. This caused the tree expansion for some words to consume up to 15 minutes. Another issue that consumes a lot of time is the path retrieval from the database. When a term needs to be fetched from the database, first its synset needs to be find. After the synset is found, the path of the synset in the hierarchical tree also needs to be found. The path is assembled starting from the terms synset traversing its parent up until the root is reached (you can think of the process traversing up the tree using Forwarding Pointers).

- Bugs in the code

  - There was missing logic in the code probably bec. these methods were never called by the GUI.
  - //TODO==========¿

What we have done?

- We managed to decouple the Code module from the GUI to be able to use each one separately .
- We fixed some bugs in the code
  - missing logic in the code ==¿code snippets
  - missing constants msh fakerha
- we modified the expansion algorithm and managed to reduce expansion time to 20% of the original running time.

How we Use it?

1. Search for both terms in the Arabic Tree in the AWN if found get theres depth and their least common ancestor.
2. if not found search database to get their synsets and expand them in the tree. This is the most expensive operation in the whole procedure due to the database request also due to the expansion operation.
3. then get the depths of both synsets and their least common ancestor
4. Finally use Wu Palmer to get the similarity using the Wu Palmer equation ========¿ref 3laha
5. Apply the semantic similarity equation ===¿reference 3laha

3.2.2.3 Wikipedia

Using the traditional approach of a controlled, designed ontology has many disadvantages beginning with the often difficult task of designing and implementing the ontology. Once that it done, it must be maintained and modified, an important process in domains where the underlying concepts are evolving rapidly. ACMs CCS, for example, undergoes periodic reorganization and redesign and yet as a classification of computer science concepts, it always seems to be out of date or even quaint. As a final problem, consider the process a person must follow in assigning ontology terms to a document. She has to be familiar with all of the possible choices or have some way to browse or search through them. She has to understand what each of the terms means, either the original meaning intended by the ontology designer or the possibly different current meaning as used by her community. Finally, she has to select the best set of terms from among the many relevant choices the ontology may present to her. The use of an implicit ontology emerging from the tagging choices of a community of individuals solves some of these problems, but also has significant disadvantages. Some of these are inherent and others are being addressed in the research community and may ultimately admit good solutions. These problems are worth addressing because the result will be an ontology that (1) represents a consensus view of a community of users and (2) is constructed and maintained by the community without cost to any organization. It remains unclear how the terms in such an ontology should be organized structurally, understood informally by end users, or mapped to a more formal ontology such as Cyc (Lenat 1995) or popular Semantic Web ontologies like FOAF (Ding et al. 2005). We are developing a system that is a blend of the two approaches based on the idea of using Wikipedia as an ontology. Specifically, each non-administrative Wikipedia page is used as a term in an ontology. These include Wikipedia articles describing individuals (Alan Turing), concepts (Emissions trading), locations (Barbados), events (collapse of the World trade Center), and categories (microbiology). Using Wikipedia as an ontology has many advantages: it is broad and fairly comprehensive, of generally high quality, constructed and maintained by tens of thousands of

users, evolves and adapts rapidly as events and knowledge change, and free and open sourced. Moreover, the meaning of any term in the ontology is easy for a person to understand from the content on the Web page. Finally, the Wikipedia pages are already linked to many existing formal ontologies through efforts like DBpedia (Auer et al. 2007) and Semantic MediaWiki (Krotzsch et al. 2006) and in commercial systems like Freebase and Powerset. The underlying concept of an article cannot be assessed by merely considering the words that appear in that article, in addition to that, finding out if two articles are conceptually related is an even more challenging problem and requires a lot of background domain knowledge, common sense as well as information about the context. Humans have the inborn ability to relate concepts semantically however it is still a very difficult problem for computers, which can be made easier by augmenting background domain knowledge for such tasks, which would certainly improve the accuracy and quality of prediction. Wikipedia proves to be an invaluable source for such background domain knowledge.

Wikipedia is a freely available online encyclopedia developed by a community of users. Wikipedia is growing exponentially and new content is being added to it daily by users around the globe. This encyclopedia comprises of millions of articles. The corpus is composed of several collections in different languages such as: English, French, German, Dutch, Chinese, Spanish, Arabic and Japanese. Each collection is a set of XML documents built using Wikipedia. Documents of the Wikipedia XML collections are organized in a hierarchy of categories defined by the authors of the articles. The Wikipedia category and article network has been studied in detail with respect to different graph properties. The Wikipedia category system is a taxonomy for arranging articles into categories and subcategories. However, this taxonomy is not a strict hierarchy or tree of categories, but allows multiple categorizations of topics simultaneously, i.e., some categories might have more than one super-category. It is shown that Wikipedias category system is a thesaurus that is collaboratively developed and used for indexing Wikipedia articles (Voss 2006). The articles within Wikipedia are inter-linked. However, these links do not impose any subcategory or super-category relationships. It has been observed that the Wikipedia article links graph generally resembles the World Wide Web graph (Zlatic et al. 2006). We downloaded the Wikipedia XML snapshot of 4 November 2006 and extracted 2,557,939 Wikipedia articles. The text of each article was indexed using the Lucene text search engine library (Gospodnetic et al. 2004) under the standard configuration. We ignored the history, talk pages, user pages, etc. We also downloaded the Wikipedia database tables in order to create the category links graph and the article links graph. Major administrative categories (e.g., All articles with unsourced statements) were identified and removed from the category links graph. Any remaining administrative categories appearing in the prediction results were excluded. We implemented three different methods for our study, which are described and discussed below. How we Use it? In the first method we use the test document or set of related documents as search query to the Lucene Wikipedia index. After getting top N matching Wikipedia articles (based on cosine similarity) for each document

in the set, we extract their Wikipedia categories and score them based on two simple scoring schemes.

- In scoring scheme one we simply count the number of times that each Wikipedia category was associated with one of the N results.
- In scoring scheme two we take into account the cosine similarity score between the test document and matching Wikipedia articles. The score for each category is the sum of the cosine similarity scores of the matching articles that are linked to the category.

3.2.3 Term Weigthing using Specificity

The following section provides more insight into term Specificity, which is the idea that some words carry more semantic content than others which is also referred to as Semantic Informativeness. Computational estimation of this quantity, term specificity, is important for various applications such as information retrieval. Here in addition to the similarity of words, we also take into account the specificity of words, so that we can give a higher weight to a semantic matching identified between two specific words (e.g. collie and sheepdog), and give less importance to the similarity measured between generic concepts (e.g. get and become). While the specificity of words is already measured to some extent by their depth in the semantic hierarchy, we are reinforcing this factor with a corpus-based measure of word specificity, based on distributional information learned from large corpora. We use two methods of computing term specificity. The first method based on modeling the rate of learning of word meaning in Latent Semantic Analysis (LSA), LSA-Specificity. The second method is TF/IDF-Specificity.

3.2.3.1 Latent Semantic Analysis:

In the following section we explain using Latent Semantic Analysis for approximating term Informativeness. Latent Semantic Analysis (LSA) is a language model that represents semantic word meaning as vectors in high-dimensional space. Word vectors are positioned in such a way that semantically-related words vectors point in similar directions or have a smaller angle / higher cosine between them. The representation is derived in an unsupervised manner, by observing occurrence patterns of words in a large corpus of natural language documents. Singular Value Decomposition (SVD) on the matrix of word/document occurrence counts is used to derive the optimal set of dimensions of the space in which all of the words can be represented as vectors. The number of dimensions is then artificially reduced to a smaller number (typically around 300) of most important dimensions, which has the effect of smoothing out incidental relationships and preserving significant ones between words. The resulting geometric space allows for straightforward representation of meaning of words and/or documents; the latter are simply a weighted

geometric composition of constituent word vectors. Similarity in meaning between a pair of words or documents can be obtained by computing the cosine between their corresponding vectors. For details of LSA, please see (Landauer et al., 2007), and others.

LSA applications almost always focus on computing the semantic similarity between words (terms). However, another property of LSA, that has a significant effect, is the term vector length. The vector length plays a very important role in many LSA calculations, in particular in giving relative weights to the word vectors that constitute a particular text passage. As (Kintsch, 2001) wrote, Intuitively, the vector length tells us how much information LSA has about this vector. [...] Words that LSA knows a lot about (because they appear frequently in the training corpus [...]) have greater vector lengths than words LSA does not know well. Function words that are used frequently in many different contexts have low vector lengths – LSA knows nothing about them and cannot tell them apart since they appear in all contexts. The metric of term Informativeness, or specificity, which we call LSAspec, is simply the ratio of LSA word vector length to the number of documents in the LSA training corpus that contain a particular word;

Missing Equation

Paper section 3.3

The value can be interpreted as the rate of vector length growth. We argue that more specific, or informative, words have the greatest rate of vector length growth; LSA learns about their meaning faster, with relatively fewer exposures. To illustrate this concept, let's look at a few examples that were obtained by controlling the number of occurrences of a particular word in the LSA training corpus.

@Mohsen: Should provide a graph for the relation of Vector lengths for some words vs. the number of documents containing those words.

3.2.3.2 Inverse Document Frequency:

The specificity of a word is determined using the inverse document frequency (IDF) introduced by (Sparck-Jones, 1972), defined as the total number of documents in the corpus divided by the total number of documents including that word. The IDF measure was selected based on previous work that theoretically proved the effectiveness of this weighting approach (Papineni, 2001). Sparck Jones (1973) defines IDF as the probability of occurrence of documents containing a particular word.

IDF equation and definition of its terms

Where D is the total number of documents in the corpus. The assumption behind it is that low frequency words tend to be rich in content, and vice versa. Theterm countin the given document is simply the number of times a giventerm appears in that document. This count is usually normalized to prevent a bias towards longer documents (which may have a higher term

count regardless of the actual importance of that term in the document) to give a measure of the importance of the termtwithin the particular documentd. One well-studied technique is to normalize the TF weights of all terms occurring in a document by the maximum TF in that document.
http://nlp.stanford.edu/IR-book/html/htmledition/maximum-tf-normalization-1.html

3.2.4 Disambiguation

Word Sense Disambiguation (WSD) is traditionally considered an AI-hard problem.A breakthrough in this field would have a significant impact on many relevant web-based applications, such as information retrieval and information extraction. This paper describes JIGSAW, a knowledge-based WSD system that attemps to disambiguate all words in a text by exploiting WordNet1 senses. The main assumption is that a specific strategy for each Part-Of-Speech (POS) is better than a single strategy. We evaluated the accuracy of JIGSAW on SemEval-2007 task 1 competition2 . This task is an application-driven one, where the application is a fixed cross-lingual information retrieval system. Participants disambiguate text by assigning WordNet synsets, then the system has to do the expansion to other languages, index the expanded documents and run the retrieval for all the languages in batch. The retrieval results are taken as a measure for the effectiveness of the disambiguation.

3.2.4.1 JIGSAW Algorithm

The goal of a WSD algorithm consists in assigninga word wi occurring in a document d with its appropriate meaning or sense s, by exploiting the context C in where wi is found. The context C for wi is defined as a set of words that precede and follow w_i .The sense s is selected from a predefined set of possibilities, usually known as sense inventory. In the proposed algorithm, the sense inventory is obtained from WordNet 1.6, according to SemEval-2007 task 1 instructions. JIGSAW is a WSD algorithm based on the idea of combining three different strategies to with sense sij . The intuition behind this algorithm disambiguate nouns, verbs, adjectives and adverbs.The main motivation behind our approach is that the effectiveness of a WSD algorithm is strongly influenced by the POS tag of the target word. An adaptation of Lesk dictionary-based WSD algorithm has been used to disambiguate adjectives and adverbs (Banerjee and Pedersen, 2002), an adaptation of the Resnik algorithm has been used to disambiguate nouns (Resnik, 1995), while the algorithm we developed for disambiguating verbs exploits the nouns in the context of the verb as well as the nouns both in the glosses and in the phrases that WordNet utilizes to describe the usage of a verb. JIGSAW takes as input a document d = w1 , w2 , . . . , wh and returns a list of WordNet synsets X = s1 , s2 , . . . ,sk in which each element si is obtained by disambiguating the target word wi based on the information obtained from WordNet about a

few immediately surrounding words. We define the context C of the target word to be a window of n words to the left and another n words to the right, for a total of 2n surrounding words. The algorithm is based on three different procedures for nouns, verbs, adverbs and adjectives, called JIGSAWnouns , JIGSAWverbs , JIGSAWothers , respectively. More details for each one of the above mentioned procedures follow. you may check code here A.2

Chapter 4

Document Clustering

Document clustering has been investigated for use in a number of different areas of text mining and information retrieval. Initially, document clustering was investigated for improving the precision or recall in information retrieval systems and as an efficient way of finding the nearest neighbors of a document. More recently, clustering has been proposed for use in browsing a collection of documents or in organizing the results returned by a search engine in response to a user's query. Document clustering has also been used to automatically generate hierarchical clusters of documents. (The automatic generation of a taxonomy of Web documents like that provided by Yahoo! [**?**]

In this work we used clustring to group similar articles together for to ease user browsing and to determine the user neighborhood to enhance the performance of the system as we don't need to use all users' profiles in the system in the recommendation algorithm to determine the recommendations to a specified user, but instead we only investigate the profiles of the user's neighborhood which are other users who are in the same cluster.

Partitioning algorithms as K-Means [**?**], itsvariants [**?**] [**?**], K-Medoids [**?**], PAM [**?**] (Partitioning AroundMedoids),BIRCH [**?**] and CLARANS [**?**], tend to use a center-based clustering objective,re-sulting in clusters of known geometrical shape,e.g. asphereoran ellipse. Grid based clustering algorithms include STING [**?**], CliQue [**?**],WaveCluster [**?**], andDenClue [**?**]. However,density-based algorithms asDBScan [**?**], DenClue [**?**], andSNN(SharedNear- est Neighbors) [**?**] have introduced new clustering criteria that do not restrict the shape of clusters, and thus are able to obtain what is called *arbitrary* shaped clusters. Yet,they depend on using a static model for clustering, which deteriorates when arbitrary densities are present in data. Chameleon [**?**], on the other hand, uses a dynamic model to be able to merge clusters of related densities together. Yet, Chameleon suffers some drawbacks.

For the reasons mentioned above we applied the model proposed in **Mitosis** [**?**] which introduces a dynamic model based on distance relatedness concepts to overcome drawbacks of related algorithms. Mitosis discovers clusters of arbitrary shapes and densities. It uses distance relatedness between patterns (in this case documents)as a measure to identify clusters of different densities. Mitosis overcomes the traditional clustering algorithms in the following points

- Combining both local and global novel distance-consistency measures in a dynamic model to cluster the data.
- Introducing a greedy clustering approach to keep up with the linear time performance of simple algorithms.
- The ability to identify outliers in the presence of clusters of different densities in the same dataset.
- The use of a parameter selection procedure to guide the clustering algorithm.
- Suitability for high dimensional data, where the discovery of arbitrary shape and density clusters reveals other relations not revealed by traditional clustering algorithms.

Section 4.1 introduces background needed to explain the algorithm.

4.1 Dynamic modeling and distance relatedness concepts

The significance of a dynamic model of clustering is first explained 4.1.1, and the theoretical background of distance relatedness is then discussed 4.1.2.

4.1.1 Dynamic vs. static models

A static model refers to using user-defined parameters for defining density thresholds without referring to the characteristics of the underlying data.DBScan, DenClue, WaveClusterandSNN, are examples of algorithms that use a static model and are unable to find clusters of arbitrary densities.For example, DBScan uses two parameters *Eps* and *Minpts*, where *Eps* specifies a threshold on range distance, and Minpts specifies the minimum number of neighbors. Apattern with a number of neighbors more than Minpts at adistance less than Eps, is considered a corepoint(*pattern*) and is used to track clusters. Chameleon,on the other hand introduced a dynamic modelfor clustering based on the following:

- The use of k-nearest neighbors(*k-NN*).

- Merging of clusters based on relative closeness and relative interconnectivity measures, weighed by user defined parameters.

4.1.2 Distance-based density

Clusters can be visually detected by recognizing relatively higher density regions in data separated by regions of relatively lower densities. It is observed that the behavior of neighborhood distances determines the degree of a cluster's density; smaller distances reflect higher densities, and viceversa. The average distances between patterns(the Minimum Spanning Tree(MST) distances) differentiate the densities of clusters in Fig. 4.1. Several other points important to build an efficient model are as follows:

- Arbitrary shaped clusters can be detected by tracking connectivity between patterns' neighborhoods.
- The distances between a pattern and its neighbors determine the density of its cluster.
- Patterns of the same cluster, should have consistent neighborhood distances to preserve a consistent density thoughout the cluster.

Distance relatedness measures: Local and global measures for distance relatedness have been separately considered for clustering. Local relatedness refers to relatedness of distances in two neighboring patterns' neighborhoods, while global relatedness refers to relatedness of distances in a group of patterns' neighborhoods to those in a neighboring group's neighborhoods. Thus global relatedness considers relating the neighborhoods of two patterns that may not be in each other's neighborhoods, but can be reached from one another through a group of neighbor patterns. Global distance relatedness does not guarantee local distance relatedness. Yet considering both local and global relatedness aims at increasing the distance consistency of each cluster, and avoiding the effect of outliers which may merge two clusters together.

4.2 Proposed distance relatedness dynamic model

The dynamic model proposed in *Mitosis* aims at maintaining the consistency of distances between patterns in each cluster.It defines a distance-relatedness clustering criterion based on the following:

- A dynamic range neighborhood that depends on the density context of the pattern.
- Measures that reflect the distance structure of the data.

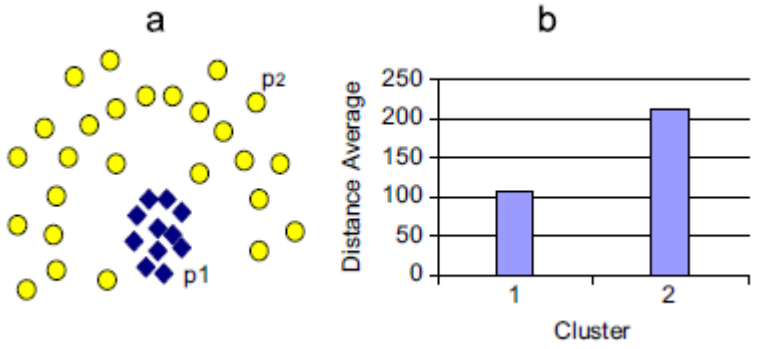


FIGURE 4.1: Distances between patterns reflect the density structure.

Mitosis distance relatedness model combines both local and global distance relatedness. It attempts to cluster the data starting from local measures and extending that to global ones in the course of the algorithm. Let the symbols x, y, p, and q denote a pattern, $d(x, y)$ denotes the distance between patterns x and y, a denotes the association between two patterns x and y, which is the tuple $(x, y, d(x, y))$, A denotes a set of associations, f and k denote user parameters, and μ denotes the average value of distances.

Local distance relatedness: Any cluster is initially formed from singleton patterns that are related with respect to their neighborhood average distance measure, and have an in-between distance related to this measure as well. This local merging criterion is a modification of Zahn'sinconsistencymodel. The rules proposed here are as follows, considering that an average local distance measure is the average of distances in a neighborhood of a pattern as given later

- The distance between two patterns is related to each of the patterns' average local distance measure, by a factor k.
- The two patterns' average local distance measures are related by factor k.

Figure 4.2 illustrates the importance of the mutual consistency between the three entities associated with the local relatedness criterion; two patterns and a nin-between distance. It shows two examples, the first example (p_1, p_2, d_1) has two patterns with distance $d_1$related to each patterns' neighborhood distances, yet the two neighborhood distance averages are not related. The second case (p_3, p_4, d_2)gives an example of two related neighborhood distance averages, but an unrelated in-between distance d_2. The two patterns do not merge in any of these cases, justifying the above proposed local criterion that imposes the mutual consistency between the two neighborhoods and the in between distance. The same rationale is used when merging two clusters, but this time considering the clusters' global characteristics.

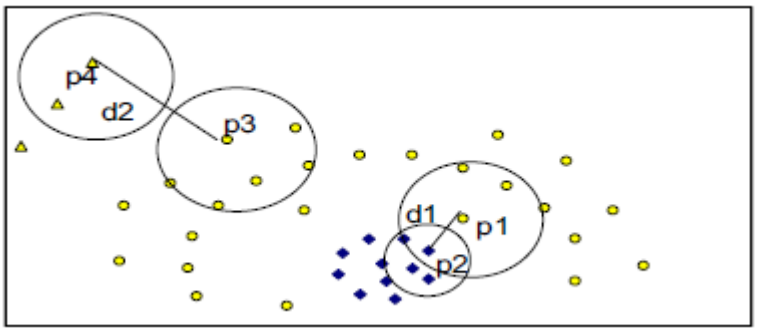


FIGURE 4.2: Effect of distance inconsistency on merging of patterns

Global distance relatedness: The property of distance consistency extends from local relations between neighborhood patterns to global relations between neighboring clusters. The global distance relatedness model follows the same rules of the local relatedness, but with replacing patterns with clusters, a pattern's local average distance with a cluster's average distance, and the distance between two singleton patterns with the distance between the two patterns in the two neighboring clusters. A cluster's average distance(only distances approved to be consistent to the cluster)is used to reflect the distance behavior inside the cluster.

4.2.1 Distance consistency definitions

The rules governing consistency of one distance to another cluster's/pattern's average distance, or two clusters'/patterns' average distance to each other are as follows. given an input parameter $k > 1$:

- A distance $d(x, y)$ is *k-consistent* to an average μ of distances if $d(x, y) < k.\mu$
- Two distance averages μ_1 and μ_2 are *k-consistent* to each other if $\mu_1 < k.\mu_2 \wedge \mu_2 < k.\mu_1$ or equivalently $\min(\mu_1, \mu_2) < k.max(\mu_1, \mu_2)$
- A distance $d(x, y)$ is *k-consistent* to two averages if $d(x, y) < k.\mu_1 \wedge d(x, y) < k.\mu_2$ i.e. $d(x, y) < k.min(\mu_1, \mu_2)$

According to the above definitions the merge criteria are proposed.

4.3 Algorithms phases

The algorithm takes as an input the dataset P, and 2 main parameters f and k. The main steps are as follows:

- Get associations:
  - For each pattern get its dynamic range nearest neighbors using parameter f, and calculate local average distance.
  - Order the associationsascendingly on distances-between patterns inalist L.
- Merge patterns in to clusters: For each association in L (visited in order), if the merging criterion is satisfiedaccording to k-, merge associated patterns/clusters, and update the clusters'measures.
- Refine clusters:
  - For each cluster, add any internal association consistent to the cluster'saverage.
  - For each cluster, calculate the Harmonic average of distances, and remove associations not consistent to this measure.

4.4 Implementation Details

Merging phase is a transitive operation, so equivalence classes must be implemented efficiently to avoid any perofrmance bottleneck. This can be achieved by using *Disjoint sets data structures*

4.4.1 Disjoint set

A disjoint-set is a collection $C = S_1, S_2, ..., S_k$ of distinct dynamic sets. Each set is identified by a member of the set, called representative. Disjoint set operations:

- MAKE-SET(x): create a new set with only x. assume x is not already in some other set.
- UNION(x,y): combine the two sets containing x and y into one new set. A new representative is selected.
- FIND-SET(x): return the representative of the set containing x.

Obviously the *MAKE-SET(x)* operation can be used at the beginning of the algorithm as each pattern will be in a cluster, and the *UNION(x,y)* will be used to marge to clusters. Next subsections will illustrate different implementations

4.4.1.1 Linked-List Implementation

Each set as a linked-list, with head and tail, and each node contains value, next node pointer and back-to-representative pointer. Figure 4.3 4.4 shows the operations of the disgoint set.

- MAKE-SET costs 1: just create a single element list.
- FIND-SET costs 1: just return back-to-representative pointer.
- UNION(x,y)
  - A simple implementation: just appends x to the end of y, updates all back-to-representative pointers in x to the head of y. Each UNION takes time linear in the xs length
  - Instead appending x to y, appending the shorter list to the longer list. Associated a length with each list, which indicates how many elements in the list.

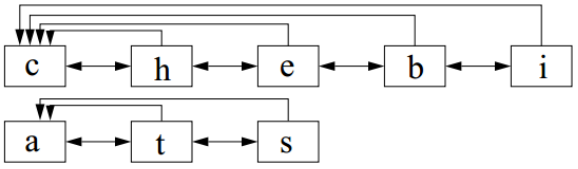


FIGURE 4.3: Two sets

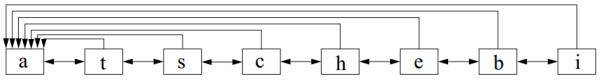


FIGURE 4.4: Union of two sets

4.4.1.2 Disjoint-set Implementation: Forests

Rooted trees, each tree is a set, root is the representative. Each node points to its parent. Root points to itself.

Three operations

- MAKE-SET(x): create a tree containing x. 1
- FIND-SET(x): follow the chain of parent pointers until to the root. height of xs tree
- UNION(x,y): let the root of one tree point to the root of the other. 1

Union by Rank and Path Compression *Union by Rank*: Each node is associated with a rank, which is the upper bound on the height of the node (i.e., the height of subtree rooted at the node), then when UNION, let the root with smaller rank point to the root with larger rank. *Path Compression*: used in FIND-SET(x) operation, make each node in the path from x to the root directly point to the root. Thus reduce the tree height.

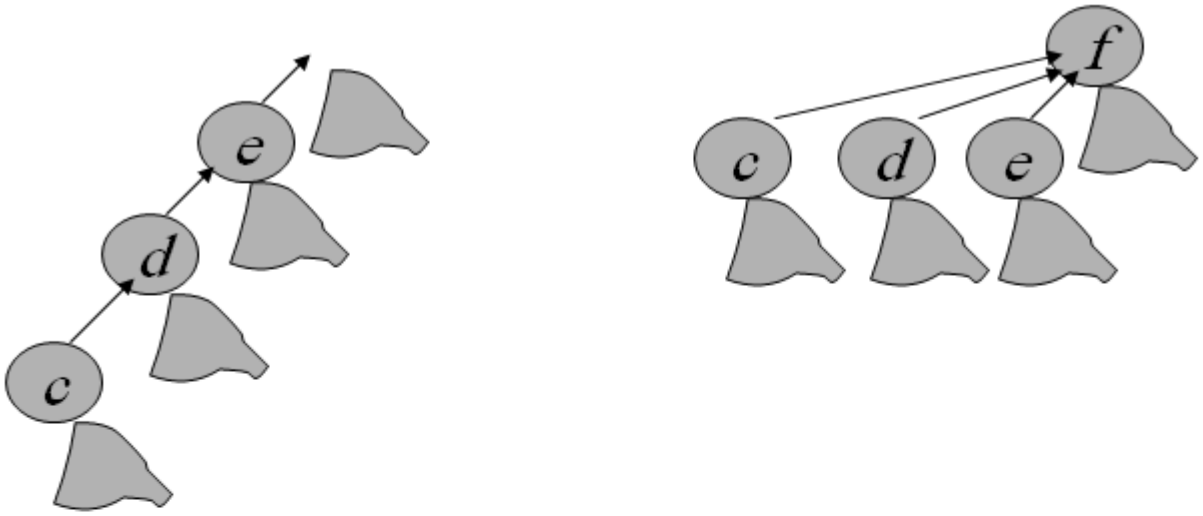


FIGURE 4.5: Path compression

Chapter 5

Recommender Systems

5.1 Introduction

Recommender systems became an important research area since the appearance of the first papers on collaborative filtering since the mid-1990s [**?**] [**?**] [**?**]. There has been much work done both in the industry and academia on developing new approaches to recommender systems over the last decade. The interest in this area still remains high because it constitutes a problem-rich research area and because of the abundance of practical applications that help users to deal with information overload and provide personalized recommendations, content and services to them. Examples of such applications include recommending books, CDs and other products at Amazon.com [**?**], movies by MovieLens [**?**], and news at VERSIFI Technologies (formerly AdaptiveInfo.com) [**?**]. Moreover, some of the vendors have incorporated recommendation capabilities into their commerce servers [**?**].

5.2 Recommendation Problem

The recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user. Intuitively, this estimation is usually based on the ratings given by this user to other items and on some other information that will be formally described below. Once we can estimate ratings for the yet unrated items, we can recommend to the user the item(s) with the highest estimated rating(s). More formally, the recommendation problem can be formulated as follows. Let C be the set of all users and let S be the set of all possible items that can be recommended, such as books, movies, or restaurants. The space S of possible items can be very large, ranging in hundreds of thousands or even millions of items in some applications, such as recommending books or CDs. Similarly, the user space can also be very large millions in some cases. Let u be a utility function that measures usefulness of item s to user c, i.e., $u : C \times S \rightarrow R$

, where R is a totally ordered set (e.g., non-negative integers or real numbers within a certain range). Then for each user $c \in C$, we want to choose such item $s' \in S$ that maximizes the user's utility. More formally

$$\forall c \in C,\ S'_c = arg_{max u}(c, s)_{s \in S} \tag{5.1}$$

In recommender systems the utility of an item is usually represented by a rating, which indicates how a particular user liked a particular item, e.g., John Doe gave the movie Harry Potter the rating of 7 (out of 10). However, as indicated earlier, in general utility can be an arbitrary function, including a profit function. Depending on the application, utility u can either be specified by the user, as is often done for the user-defined ratings, or is computed by the application, as can be the case for a profit-based utility function. Each element of the user space C can be defined with a profile that includes various user characteristics, such as age, gender, income, marital status, etc. In the simplest case, the profile can contain only a single (unique) element, such as User ID. Similarly, each element of the item space S is defined with a set of characteristics. For example, in a movie recommendation application, where S is a collection of movies, each movie can be represented not only by its ID, but also by its title, genre, director, year of release, leading actors, etc.

The central problem of recommender systems lies in that utility u is usually not defined on the whole $C \times S$ space, but only on some subset of it. This means u needs to be extrapolated to the whole space $C \times S$. In recommender systems, utility is typically represented by ratings and is initially defined only on the items previously rated by the users. For example, in a movie recommendation application (such as the one at MovieLens.org), users initially rate some subset of movies that they have already seen. An example of a user-item rating matrix for a movie recommendation application is presented in Table 5.1, where ratings are specified on the scale of 1 to 5. The ϕ symbol for some of the ratings in Table 5.1 means that the users have not rated the corresponding movies. Therefore, the recommendation engine should be able to estimate (predict) the ratings of the non-rated movie/user combinations and issue appropriate recommendations based on these predictions.

TABLE 5.1: A fragment of a rating matrix for a movie recommender system

	K PAX	Life of Brain	Memento	Notorious
Alice	4	3	2	4
Bob	ϕ	4	5	5
Cindy	2	2	4	ϕ
David	3	ϕ	5	2

Extrapolations from known to unknown ratings are usually done by:

- Specifying heuristics that define the utility function and empirically validating its performance

- Estimating the utility function that optimizes certain performance criterion, such as the mean square error.

Once the unknown ratings are estimated, actual recommendations of an item to a user are made by selecting the highest rating among all the estimated ratings for that user, according to formula 5.1. Alternatively, we can recommend N best items to a user or a set of users to an item. The new ratings of the not-yet-rated items can be estimated in many different ways using the methods from machine learning, approximation theory and various heuristics. Recommender systems are usually classified according to their approach to rating estimation.

Recommender systems are usually classified into the following categories, based on how recommendations are made:

- Content-based recommendations: the user is recommended items similar to the ones the user preferred in the past
- Collaborative recommendations: the user is recommended items that people with similar tastes and preferences liked in the past
- Hybrid approaches: these methods combine collaborative and content-based methods

5.3 Content-Based Methods

In content-based recommendation methods, the utility $u(c, s)$ of item s for user c is estimated based on the utilities $u(c, s_i)$ assigned by user c to items $s_i \in S$ that are *similar* to item s. For example, in a movie recommendation application, in order to recommend movies to user c, the content-based recommender system tries to understand the commonalities among the movies user c has rated highly in the past (specific actors, directors, genres, subject matter, etc.). Then, only the movies that have a high degree of similarity to whatever users preferences are would get recommended.

The content-based approach to recommendation has its roots in information retrieval [**?**] [**?**] and information filtering [**?**] research. Because of the significant and early advancements made by the information retrieval and filtering communities and because of the importance of several text-based applications, many current content-based systems focus on recommending items containing textual information, such as documents, Web sites (URLs), and Usenet news messages. The improvement over the traditional information retrieval approaches comes from the use of user profiles that contain information about users tastes, preferences, and needs. The profiling information can be elicited from users explicitly, e.g., through questionnaires, or implicitly learned from their transactional behavior over time.

More formally, let Content(s) be an item profile, i.e., a set of attributes characterizing item s. It is usually computed by extracting a set of features from item s (its content) and is used to determine appropriateness of the item for recommendation purposes. Since, as mentioned earlier, content-based systems are designed mostly to recommend text-based items, the content in these systems is usually described with keywords. For example, a content-based component of the Fab system [**?**], which recommends Web pages to users, represents Web page content with the 100 most important words. Similarly, the Syskill & Webert system [**?**] represents documents with the 128 most informative words. The "importance" (or "informativeness") of word k_i in document d_j is determined with some weighting measure w_{ij} that can be defined in several different ways.

One of the best-known measures for specifying keyword weights in Information Retrieval is the term frequency/inverse document frequency (TF-IDF) measure [89] that is defined as follows. Assume that N is the total number of documents that can be recommended to users and that keyword ki appears in ni of them. Moreover, assume that $f_{i,j}$ is the number of times keyword k_i appears in document d_j. Then $TF_{i,j}$, the term frequency (or normalized frequency) of keyword k_i in document d_j, is defined as

$$TF_{i,j} = \frac{f_{i,j}}{max_z f_{z,j}} \tag{5.2}$$

Where the maximum is computed over the frequencies $f_{z,j}$ of all keywords k_z that appear in the document d_j. However, keywords that appear in many documents are not useful in distinguishing between a relevant document and a non-relevant one. Therefore, the measure of inverse document frequency (IDF_i) is often used in combination with simple term frequency ($TF_{i,j}$). The inverse document frequency for keyword k_i is usually defined as

$$IDF_i = log(\frac{N}{n_i}) \tag{5.3}$$

Then the TF-IDF weight for keyword k_i in document d_j is defined as

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{5.4}$$

and the content of document d_j is defined as Content(d_j) = (w_{1j}, w_{kj}).

As stated earlier, content-based systems recommend items similar to those that a user liked in the past [**?**] [**?**] [**?**]. In particular, various candidate items are compared with items previously rated by the user, and the best-matching item(s) are recommended. More formally, let ContentBasedProfile(c) be the profile of user c containing tastes and preferences of this user. These profiles are obtained by analyzing the content of the items previously seen and rated by the

user and are usually constructed using keyword analysis techniques from information retrieval. For example, ContentBasedProfile(c) can be defined as a vector of weights $(w_{c_1}, ..., w_{c_k})$, where each weight w_{c_i} denotes the importance of keyword k_i to user c and can be computed from individually rated content vectors using a variety of techniques. For example, some averaging approach, such as Rocchio algorithm [**?**], can be used to compute ContentBasedProfile(c) as an *average* vector from an individual content vectors [**?**] [**?**]. On the other hand, [**?**] use a Bayesian classifier in order to estimate the probability that a document is liked. The Winnow algorithm [**?**] has also been shown to work well for this purpose, especially in the situations where there are many possible features [**?**]. In content-based systems, the utility function $u(c, s)$ is usually defined as:

$$u(c, s) = score(ContentBasedProfile(c), Content(s)) \tag{5.5}$$

Using the above-mentioned information retrieval-based paradigm of recommending Web pages, Web site URLs, or Usenet news messages, both ContentBasedProfile(c) of user c and Content(s) of document s can be represented as TF-IDF vectors $\vec{W}_c$ and $\vec{W}_S$ of keyword weights. Moreover, utility function $u(c, s)$ is usually represented in information retrieval literature by some scoring heuristic defined in terms of vectors $\vec{W}_c$ and $\vec{W}_S$, such as cosine similarity measure [**?**] [**?**]:

$$u(c, s) = cos(\vec{W}_c, \vec{W}_S) = \frac{\vec{W}_c \cdot \vec{W}_S}{\|\vec{W}_c\|_2 \times \|\vec{W}_S\|_2} = \frac{\sum_{i=1}^{k} w_{i,c} \cdot w_{i,s}}{\sqrt{\sum_{i=1}^{k} w_{i,c}^2} \sqrt{\sum_{i=1}^{k} w_{i,s}^2}} \tag{5.6}$$

where K is the total number of keywords in the system. For example, if user c reads many online articles on the topic of bioinformatics, then content-based recommendation techniques will be able to recommend other bioinformatics articles to user c. This is the case, because these articles will have more bioinformatics-related terms (e.g., genome, sequencing, proteomics) than articles on other topics, and, therefore, ContentBasedProfile(c), as defined by vector $\vec{W}_c$, will represent such terms k_i with high weights w_{ic}. Consequently, a recommender system using the cosine or a related similarity measure will assign higher utility u(c, s) to those articles s that have high-weighted bioinformatics terms in $\vec{w}_s$ and lower utility to the ones where bioinformatics terms are weighted less. Besides the traditional heuristics that are based mostly on information retrieval methods, other techniques for content-based recommendation have also been used, such as Bayesian classifiers [**?**] [**?**] and various machine learning techniques, including clustering, decision trees, and artificial neural networks [**?**]. These techniques differ from information retrieval-based approaches in that they calculate utility predictions based not on a heuristic formula, such as a cosine similarity measure, but rather are based on a model learned from the underlying data using statistical learning and machine learning techniques. For example, based

on a set of Web pages that were rated as relevant or irrelevant by the user, [**?**] use the nave Bayesian classifier [**?**] to classify unrated Web pages. More specifically, the nave Bayesian classifier is used to estimate the following probability that page p_j belongs to a certain class C_i (e.g., relevant or irrelevant) given the set of keywords $k_{1,j}, ..., kn, j$ on that page:

$$P(C_i|k_{1,j}\&...\&K_{n,j}) \tag{5.7}$$

While not explicitly dealing with providing recommendations, the text retrieval community has contributed several techniques that are being used in content-based recommender systems. One example of such technique would be the research on adaptive filtering [**?**] [**?**], which focuses on becoming more accurate at identifying relevant documents incrementally, by observing the documents one-by-one in a continuous document stream. Another example would be the work on threshold setting [**?**] [**?**], which focuses on determining the extent to which documents should match a given query in order to be relevant to the user.

5.3.1 Limitation

5.3.1.1 Limited Content Analysis

Content-based techniques are limited by the features that are explicitly associated with the objects that these systems recommend. Therefore, in order to have a sufficient set of features, the content must either be in a form that can be parsed automatically by a computer (e.g., text), or the features should be assigned to items manually. While information retrieval techniques work well in extracting features from text documents, some other domains have an inherent problem with automatic feature extraction. For example, automatic feature extraction methods are much harder to apply to the multimedia data, e.g., graphical images, audio and video streams. Moreover, it is often not practical to assign attributes by hand due to limitations of resources [**?**]. Another problem with limited content analysis is that, if two different items are represented by the same set of features, they are indistinguishable. Therefore, since text-based documents are usually represented by their most important keywords, content-based systems cannot distinguish between a well-written article and a badly written one, if they happen to use the same terms [**?**].

5.3.1.2 Over-Specialization

When the system can only recommend items that score highly against a users profile, the user is limited to being recommended items similar to those already rated. For example, a person with no experience with Greek cuisine would never receive a recommendation for even the greatest

Greek restaurant in town. This problem, which has also been studied in other domains, is often addressed by introducing some randomness. For example, the use of genetic algorithms has been proposed as a possible solution in the context of information filtering [**?**]. In addition, the problem with over-specialization is not only that the content-based systems cannot recommend items that are different from anything the user has seen before. In certain cases, items should not be recommended if they are too similar to something the user has already seen, such as different news article describing the same event. Therefore, some content based recommender systems, such as DailyLearner [**?**], filter out items not only if they are too different from users preferences, but also if they are too similar to something the user has seen before. Furthermore, [**?**] provide a set of five redundancy measures to evaluate whether a document that is deemed to be relevant contains some novel information as well. In summary, the diversity of recommendations is often a desirable feature in recommender systems. Ideally, the user should be presented with a range of options and not with a homogeneous set of alternatives. For example, it is not necessarily a good idea to recommend all movies by Woody Allen to a user who liked one of them.

5.3.1.3 New User Problem

The user has to rate a sufficient number of items before a content-based recommender system can really understand users preferences and present the user with reliable recommendations. Therefore, a new user, having very few ratings, would not be able to get accurate recommendations.

5.4 Collaborative Methods

Unlike content-based recommendation methods, collaborative recommender systems (or collaborative filtering systems) try to predict the utility of items for a particular user based on the items previously rated by other users. More formally, the utility $u(c, s)$ of item s for user c is estimated based on the utilities u(cj, s) assigned to item s by those users cj?C who are "similar" to user c. For example, in a movie recommendation application, in order to recommend movies to user c, the collaborative recommender system tries to find the "peers" of user c, i.e., other users that have similar tastes in movies (rate the same movies similarly). Then, only the movies that are most liked by the peers of user c would get recommended. There have been many collaborative systems developed in the academia and the industry.It can be argued that the Grundy system [87] was the first recommender system, which proposed to use stereotypes as a mechanism for building models of users based on a limited amount of information on each individual user. Using stereotypes, the Grundy system would build individual user models and use them to recommend relevant books to each user. Later on, the Tapestry system relied on each user to identify like-minded users manually [38]. GroupLens [53, 86], Video Recommender [45], and Ringo [97] were the first systems to use collaborative filtering algorithms to automate prediction. Other

examples of collaborative recommender systems include the book recommendation system from Amazon.com, the PHOAKS system that helps people find relevant information on the WWW [103], and the Jester system that recommends jokes [39].

5.4.1 Collaborative recommendations algorithms

According to [15], algorithms for collaborative recommendations can be grouped into two general classes: memory-based (or heuristic-based) and model-based. Memory-based algorithms [15, 27, 72, 86, 97] essentially are heuristics that make rating predictions based on the entire collection of previously rated items by the users. That is, the value of the unknown rating rc,s for user c and item s is usually computed as an aggregate of the ratings of some other (usually the N most similar) users for the same item s:

$$rc, s = aggr_{c' \in \hat{C}} r_{c',s} \tag{5.8}$$

where C denotes the set of N users that are the most similar to user c and who have rated item s (N can range anywhere from 1 to the number of all users). Some examples of the aggregation function are:

$$r_{c,s} = \frac{1}{N} \sum_{c' \in \hat{C}} r_{c',s} \tag{5.9}$$

$$r_{c,s} = k \sum_{c' \in \hat{C}} sim(c, c') \times r_{c',s} \tag{5.10}$$

$$r_{c,s} = \bar{r_c} + k \sum_{c' \in \hat{C}} sim(c, c') \times (r_{c',s} - \bar{r_{c'}}) \tag{5.11}$$

where multiplier k serves as a normalizing factor and is usually selected as $k = \frac{1}{\sum_{c' \in \hat{C}} |sim(c,c')|}$ and , and where the average rating of user c, rc in (10c) is defined as

$$\bar{r_c} = \frac{1}{|s_c|} \sum_{s \in S_c} r_{c,s} \; where \, S_c = \{s \in S r_{c,s} \neq \phi\} \tag{5.12}$$

In the simplest case, the aggregation can be a simple average, as defined by expression (10a).However, the most common aggregation approach is to use the weighted sum, shown in (10b).The similarity measure between the users c and c, sim(c, c), is essentially a distance measure and is used as a weight, i.e., the more similar users c and c are, the more weight rating rc,s will carry in the prediction of rc,s. Note that sim(x,y) is a heuristic artifact that is introduced in order to be able to differentiate between levels of user similarity (i.e., to be able to find a set of closest peers or nearest neighbors for each user) and at the same time simplify the rating estimation procedure. As shown in (10b), different recommendation applications can use their own user similarity measure, as long as the calculations are normalized using the normalizing factor k, as shown above.

The two most commonly used similarity measures will be described below. One problem with using the weighted sum, as in (10b), is that it does not take into account the fact that different users may use the rating scale differently. The adjusted weighted sum, shown in (10c), has been widely used to address this limitation. In this approach, instead of using the absolute values of ratings, the weighted sum uses their deviations from the average rating of the corresponding user.

5.4.2 Computing similarity between users

Various approaches have been used to compute the similarity sim(c,c') between users in collaborative recommender systems. In most of these approaches, the similarity between two users is based on their ratings of items that both users have rated. The two most popular approaches are correlation and cosine-based. To present them, let S_{xy} be the set of all items co-rated by both users x and y, i.e. $s_{xy} = \{s \in S | r_{x,s} \neq \phi \,\&\, r_{y,s} \neq \phi\}$. In collaborative recommender systems Sxy is used mainly as an intermediate result for calculating the "nearest neighbors" of user x and is often computed in a straightforward manner, i.e., by computing the intersection of sets S_x and S_y. However, some methods, such as the graph-theoretic approach to collaborative filtering [4], can determine the nearest neighbors of x without computing S_{xy} for all users y.

5.4.2.1 Correlation based similarity

In the correlation-based approach, the Pearson correlation coefficient is used to measure the similarity [86, 97]:

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r_x})(r_{y,s} - \bar{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r_x})^2 \sum_{s \in S_{xy}} (r_{y,s} - \bar{r_y})^2}} \tag{5.13}$$

5.4.2.2 Cosine based similarity

In the cosine-based approach [15, 91], the two users x and y are treated as two vectors in m-dimensional space, where $m = |S_{xy}|$. Then, the similarity between two vectors can be measured by computing the cosine of the angle between them:

$$sim(x,y) = cos(\vec{x},\vec{y}) = \frac{\vec{x}.\vec{y}}{\|\vec{x}\|_2 \times \|\vec{y}\|_2} = \frac{\sum_{i=1}^{k} r_{x,s} r_{y,s}}{\sqrt{\sum_{s \in S_{xy}} r_{x,s}^2} \sqrt{\sum_{s \in S_{xy}} r_{y,s}^2}} \tag{5.14}$$

where $\vec{x}.\vec{y}$ denotes the dot-product between the vectors $\vec{x}\, and\, \vec{y}$. Still another approach to measuring similarity between users uses the mean squared difference measure and is described in [97]. Note that different recommender systems may take different approaches in order to implement the

user similarity calculations and rating estimations as efficiently as possible. Many performance-improving modifications, such as default voting, inverse user frequency, case amplification [15], and weighted-majority prediction [27, 72], have been proposed as extensions to these standard correlation-based and cosine-based techniques.

Also, while the above techniques traditionally have been used to compute similarities between users, [91] proposed to use the same correlation-based and cosine-based techniques to compute similarities between items instead and obtain the ratings from them. This idea has been further extended in [29] for top-N item recommendations. In addition, [29, 91] present empirical evidence that item-based algorithms can provide better computational performance than traditional user-based collaborative methods, while at the same time providing comparable or better quality than the best available user-based algorithms. In contrast to memory-based methods, model-based algorithms [11, 15, 37, 39, 47, 64, 75, 105] use the collection of ratings to learn a model, which is then used to make rating predictions. For example, [15] proposes a probabilistic approach to collaborative filtering, and it is assumed that rating values are integers between 0 and n, and the probability expression is the probability that user c will give a particular rating to item s given that users ratings of the previously rated items. To estimate this probability, [15] proposes two alternative probabilistic models: **cluster models and Bayesian networks**.

5.4.3 Cluster models

Like-minded users are clustered into classes. Given the users class membership, the user ratings are assumed to be independent, i.e., the model structure is that of a nave Bayesian model. The number of classes and the parameters of the model are learned from the data. One limitation of this approach is that each user can be clustered into a single cluster, whereas some recommendation applications may benefit from the ability to cluster users into several categories at once. For example, in a book recommendation application, a user may be interested in one topic (e.g., programming) for work purposes and a completely different topic (e.g., fishing) for leisure.

5.4.4 Bayesian networks

Represents each item in the domain as a node in a Bayesian network, where the states of each node correspond to the possible rating values for each item. Both the structure of the network and the conditional probabilities are learned from the data.

5.4.5 Default Voting

an extension to the memory-based approaches described above. It was observed that whenever there are relatively few user-specified ratings, these methods would not work well in computing

similarity between users x and y since the similarity measure is based on the intersection of the itemsets, i.e., sets of items rated by both users x and y .It was empirically shown that the rating prediction accuracy could improve if we assume some default rating value for the missing ratings [15].

5.4.6 Limitation

5.4.6.1 New user problem

It is the same problem as with content-based systems. In order to make accurate recommendations, the system must first learn the users preferences from the ratings that the user makes. Several techniques have been proposed to address this problem. Most of them use hybrid recommendation approach, which combines content-based and collaborative techniques. The next section describes hybrid recommender systems in more detail. An alternative approach is presented in [83, 109], where various techniques are explored for determining the best (i.e., most informative to a recommender system) items for a new user to rate. These techniques use strategies that are based on item popularity, item entropy, user personalization, and combinations of the above [83, 109].

5.4.6.2 New item problem

New items are added regularly to recommender systems. Collaborative systems rely solely on users preferences to make recommendations. Therefore, until the new item is rated by a substantial number of users, the recommender system would not be able to recommend it.

5.4.6.3 Sparsity

In any recommender system, the number of ratings already obtained is usually very small compared to the number of ratings that need to be predicted. Effective prediction of ratings from a small number of examples is important. Also, the success of the collaborative recommender system depends on the availability of a critical mass of users. For example, in the movie recommendation system there may be many movies that have been rated only by few people and these movies would be recommended very rarely, even if those few users gave high ratings to them. Also, for the user whose tastes are unusual compared to the rest of the population there will not be any other users who are particularly similar, leading to poor recommendations [8]. One way to overcome the problem of rating sparsity is to use user profile information when calculating user similarity. That is, two users could be considered similar not only if they rated the same movies similarly, but also if they belong to the same demographic segment. For example, [76]

uses gender, age, area code, education, and employment information of users in the restaurant recommendation application. This extension of traditional collaborative filtering techniques is sometimes called demographic filtering [76]. Another approach that also explores similarities among users has been proposed in [49], where the sparsity problem is addressed by applying associative retrieval framework and related spreading activation algorithms to explore transitive associations among consumers through their past transactions and feedback. A different approach for dealing with sparse rating matrices was used in [11, 90], where a dimensionality reduction technique, Singular Value Decomposition (SVD), was used to reduce dimensionality of sparse ratings matrices. SVD is a well-known method for matrix factorization that provides the best lower rank approximations of the original matrix [90].

5.5 Hybrid Methods

Several recommendation systems use a hybrid approach by combining collaborative and content-based methods, which helps to avoid certain limitations of content-based and collaborative systems. Different ways to combine collaborative and content-based methods into a hybrid recommender system can be classified as follows:

1 Implementing collaborative and content-based methods separately and combining their predictions

2 incorporating some content-based characteristics into a collaborative approach

3 incorporating some collaborative characteristics into a content-based approach

4 constructing a general unifying model that incorporates both content-based and collaborative characteristics.

5.5.1 Combining separate recommenders

One way to build hybrid recommender systems is to implement separate collaborative and content-based systems. Then we can have two different scenarios. First, we can combine the outputs (ratings) obtained from individual recommender systems into one final recommendation using either a linear combination of ratings [21] or a voting scheme [76]. Alternatively, we can use one of the individual recommenders, at any given moment choosing to use the one that is "better" than others based on some recommendation "quality" metric For example, the DailyLearner system [13] selects the recommender system that can give the recommendation with the higher level of confidence, while [104] chooses the one whose recommendation is more consistent with past ratings of the user.

5.5.2 Adding content-based characteristics to collaborative models

Several hybrid recommender systems, including Fab [8] and the collaboration via content approach, described in [76], are based on traditional collaborative techniques but also maintain the content-based profiles for each user. These content-based profiles, and not the commonly rated items, are then used to calculate the similarity between two users. As mentioned in [76], this allows to overcome some sparsity-related problems of a purely collaborative approach, since typically not many pairs of users will have a significant number of commonly rated items. Another benefit of this approach is that users can be recommended an item not only when this item is rated highly by users with similar profiles, but also directly, i.e., when this item scores highly against the users profile [8]. [40] employs a somewhat similar approach in using the variety of different filterbots specialized content-analysis agents that act as additional participants in a collaborative filtering community. As a result, the users whose ratings agree with some of the filterbots ratings would be able to receive better recommendations [40]. Similarly, [65] uses a collaborative approach where the traditional users ratings vector is augmented with additional ratings, which are calculated using a pure content-based predictor.

5.5.3 Adding collaborative characteristics to content-based models

The most popular approach in this category is to use some dimensionality reduction technique on a group of content-based profiles. For example, [100] use latent semantic indexing (LSI) to create a collaborative view of a collection of user profiles, where user profiles are represented by term vectors resulting in a performance improvement.

5.5.4 Developing a single unifying recommendation model

Many researchers have followed this approach in recent years. For instance, [9] propose to use content-based and collaborative characteristics (e.g., the age or gender of users or the genre of movies) in a single rule-based classifier. [80] And [94] propose a unified probabilistic method for combining collaborative and content-based recommendations, which is based on the probabilistic latent semantic analysis [46]. Yet another approach is proposed by [25] and [5], where Bayesian mixed-effects regression models are used that employ Markov chain Monte Carlo methods for parameter estimation and prediction.

5.6 Demographic-based Methods

Demographic information can be used to identify the types of users that like a certain object. For example, Table 3 shows information on the age, gender, education, etc. of people that rated

a restaurant together with their rating of the restaurant. One might expect to learn the type of person that likes a certain restaurant. Similarly, Lifestyle Finder (Kruwlich 1997) attempts to identify one of 62 pre-existing clusters to which a user belongs and to tailor recommendations to user based upon information about others in the cluster. Obtaining demographic information can be difficult. Lifestyle Finder enters into dialog with the user to help categorize the user. In this work, we consider an alternative approach to obtaining demographic information in which we minimize the effort required to obtain information about users by leveraging the work the user has already expended in creating a home page on the World Wide Web . Therefore, instead of using approaches to learning from a structured database, we use text classification to classify users. The positive examples are the HTML home pages of users that like a particular restaurant and the negative examples are the HTML home pages of users that do not like that restaurant. The Winnow algorithm can be used to learn the characteristics of home pages associated with users that like a particular restaurant. While it might be preferable to obtain (or extract) structured information about users, we argue that there is a trade-off between the quality of the demographic information obtained and the amount of effort required to obtain it.

5.7 Collaboration via content

Collaborative methods look for similarities between users to make predictions. Typically, the pattern of ratings of individual users is used to determine similarity. Such a correlation is most meaningful when there are many objects rated in common between users. The content-based profile of each user is exploited to detect similarities among users. Recall that the users content-based profile contains weights for the terms that indicate that a user will like an object. Before calculating the similarity between profiles, terms that are irrelevant are deleted from each profile. This avoids considering two users to be very similar if they share a large number of terms that are irrelevant and have low weights. When computing Pearsons r between two profiles, any word in one profile but not another is treated as having a weight of 0 in the other profile. As in collaborative filtering, the prediction made is determined by a weighted average of all users predictions for that item using the correlation between profiles as the weight.

5.8 Building user Profile

For the content-based approach which we are taking, there are four essential requirements: The assumption underlying content-based systems is that the content (rather than appearance, interactivity, speed of loading, etc.) of a page is what determines the user's interest. Now we make the further assumption that we can represent the content of a page purely by considering the

TABLE 5.2: The four essential requirements

Requirement	Description
w	A representation of a web page.
m	A representation of the user's interests.
r(w,m)	A function to determine the pertinence of a web page given a user's interests.
u(w,m,s)	A function returning an updated user profile m given the user's feedback s on a page w.

words contained in the text. We ignore all mark-up tags, images and other multimedia information. The vector-space model of information retrieval (Salton McGill 1983) provides us with an appropriate representation for documents based on their constituent words. This model has been used and studied extensively, forms the basis for commercial web search systems and has been shown to be competitive with alternative IR methods (Harman 1995).

In this model documents and queries are represented as vectors. Assume some dictionary vector d, where each element d_i is a word. Each document then has a vector w, where element w_i is the weight of word d_i for that document. If the document does not contain d_i then $w_i = 0$. In our formulation we reduce words to their stems, ignore words from a stop list words, and calculate a TFIDF weight.

In an attempt to avoid over-fitting, and reduce memory and communications load, we use just the 100 highest-weighted words from any document. Recent experiments described in (Pazzani, Muramatsu, & Billsus 1996) have shown that using too many words leads to a decrease in performance when classifying web pages using supervised learning methods. So, the top N informative words are chosen using TFIDF weighting to determine them.

Once the top N words have been picked we normalize w to be of unit length, to allow comparisons between documents of different lengths. This vector representation is used both for pages, as explained, and for the model of the user's interests, the user profile m (which corresponds to the query in a retrospective IR system). In order to measure how well a page w matches a profile m, we use a variant of the standard IR cosine measure:

$$r(w, m) = w.m \tag{5.15}$$

Updating m also corresponds to a normal operation in retrospective IR: relevance feedback (Rocchio 1971).We use a simple update rule:

$$u(w, m, s) = m + sw \tag{5.16}$$

where s is the user's score for page w (we translate the scale shown in following Figure to the integers 3, 2, 1, 0, -1,-2, -3).

We assume that a user's interests change over time, and furthermore that this happens in real time, regardless of the number of recommendations requested per day. This process is modeled using a simple decay function-each night all the weights in the profiles are multiplied by 0.97.

5.9 User's Feedback

5.10 Collaboration via content

5.10.1 Explicit feedback

Explicit feedback is not a good choice for two reasons. Users will have to rate items frequently and will be very reluctant in doing so as their ratings expire quickly.

5.10.2 Implicit feedback

Acquiring feedback by observing the users actions is favorable. A user that selects and reads a document for a certain amount of time provides a strong indication that the document contains information a user is interested in. After such action the documents is therefore classified as a positive example. Finding negative examples is much more problematic. Users ignoring links to documents could be seen as a clue. This action does not provide strong evidence however as users might not have noticed the link or visit the document later. Another possible clue is a document that is being read for a very short time but this could be caused by the fact that the document is similar to what the user has already seen although the topic of the document is still interesting to the user. Classifying documents as negative based on weak assumptions leads to much noise in the training data and may result in inaccurate predictions. Negative examples will therefore not be used. In short, the user model has to be dynamic and learned from positive examples only.

5.11 Using LSA in Recommendation

Latent semantic analysis provides a vector of a predefined length (the number of documents used in building the term versus document matrix) for representing documents. So, we proposed using LSA vector representation to represent both documents and users profile.

5.11.1 Advantage of using LSA

- The length of both user profile vector and document is the same

- Calculating correlation between users is done in constant time
- Updating user profile is done in constant time just by adding the vector of the document multiplied by user feedback to the user vector

Chapter 6

Chapter 6

Tools

6.1 AWN

Arabic WordNet is the Arabic analogue to the widely used WordNet for the English language. The Arabic WordNet (AWN) is a lexical database of the Arabic language following the development process of Princeton English WordNet and Euro WordNet. It utilizes the Suggested Upper Merged Ontology as an interlingua to link Arabic WordNet to previously developed wordnets. Christiane Fellbaum at Princeton was the project lead. The project was sponsored by DOI/REFLEX.
From http://www.globalwordnet.org/AWN/DataSpec.html you can get the XML data exchange specifications of the database. AWN contains about 11,000 synsets (including 1,000 NE).

There are several different ways for accessing the database:

1 The browser package (available at http://sourceforge.net/projects/awnbrowser/) includes the AWN data and Princeton WN2.0 mappings in a relational database. You can use the export facilities to export the data as XML or CSV to taylor them to your needs .

2 The database can also be downloaded in XML format (linked to Princeton WN 2.0) from http://nlp.lsi.upc.edu/awn/get_bd.php

3 A set of basic python functions for accessing the database can be obtained from: http://nlp.lsi.upc.edu/awn/AWNDatabaseManagement.py.gz

Functionality:

- AWN Browser: Browsing the database
- AWN can be downloaded in XML format and access its content be directly used at developers' will.

Technology:
Java, Perl, MySQL
Innovation:
One of the most important lexical resources for Arabic language.

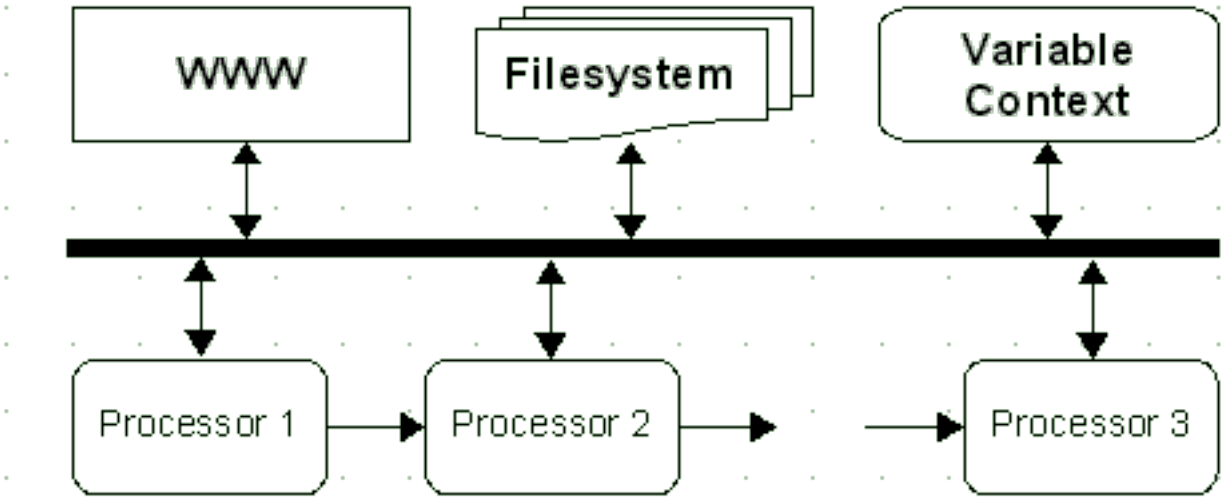
6.2 Crawlers

we used Web-Harvest for web pages crawling and retrieving the desired data.Its an Open Source Web Data Extraction tool written in Java. It offers a way to collect desired Web pages and extract useful data from them. In order to do that, it leverages well established techniques and technologies for text/xml manipulation such as XSLT, XQuery and Regular Expressions. Web-Harvest mainly focuses on HTML/XML based web sites which still make vast majority of the Web content. On the other hand, it could be easily supplemented by custom Java libraries in order to augment its extraction capabilities. Process of extracting data from Web pages is also referred as Web Scraping or Web Data Mining. World Wide Web, as the largest database, often contains various data that we would like to consume for our needs. The problem is that this data is in most cases mixed together with formatting code - that way making human-friendly, but not machine-friendly content. Doing manual copy-paste is error prone, tedious and sometimes even impossible. Web software designers usually discuss how to make clean separation between content and style, using various frameworks and design patterns in order to achieve that. Anyway, some kind of merge occurs usually at the server side, so that the bunch of HTML is delivered to the web client. Web-Harvest is distributed under BSD License. It gives the freedom for anyone to use, explore, modify, and distribute Web-Harvest, but without any warranty.

6.2.1 Basic concept

The main goal behind Web-Harvest is to empower the usage of already existing extraction technologies. Its purpose is not to propose a new method, but to provide a way to easily use and

combine the existing ones. Web-Harvest offers the set of processors for data handling and control flow. Each processor can be regarded as a function - it has zero or more input parameters and gives a result after execution. Processors could be combined in a pipeline, making the chain of execution. For easier manipulation and data reuse Web-Harvest provides variable context where named variables are stored. The following diagram describes one pipeline execution:



The result of extraction could be available in files created during execution or from the variable context if Web-Harvest is programmatically used. Configuration language

Every extraction process is defined in one or more configuration files, using simple XML-based language. Each processor is described by specific XML element or structure of XML elements. For the illustration, here is presented an example of configuration file: you may check the XML configuration file here A.1

This configuration contains two pipelines. The first pipeline performs the following steps:

1 HTML content at http://news.bbc.co.uk is downloaded

2 HTML cleaning is performed on downloaded content producing XHTML,

3 XPath expression is searched for, giving URL sequence of page images,

4 New variable named "urlList" is defined containing sequence of image URLs.

The second pipeline uses result of the previous execution in order to collect all page images:

1 Loop processor iterates over URL sequence and for every item:

2 Downloads image at current URL,

3 Stores the image on the file system.

6.3 Corpus

We used two corpora one of them is Al Jazeera, which contains 4462 article from various categories like Arts,culture,Economy, International, locals, Medical, Society, Sport. The other corpus is upto-date corpus extracted using crawling tool with predefined XML files from specific News agency Like: Reuters, CNN, BBC, Al Arabiya, Youm7.This corpus was around 160 articles from many categories.

6.4 LSA Tools

6.4.1 Python

6.4.1.1 Description

Python is a remarkably powerful dynamic programming language that is used in a wide variety ofapplication domains. Python is often compared to Tcl, Perl,Ruby, Scheme or Java. Some of its key distinguishing features include:

- very clear, readable syntax
- strong introspection capabilities
- intuitive object orientation
- natural expression of procedural code
- full modularity, supporting hierarchical packages
- exception-based error handling
- very high level dynamic data types
- extensive standard libraries and third party modules for virtually every task
- extensions and modules easily written in C, C++ (or Java for Jython, or .NET languages for IronPython)
- embeddable within applications as a scripting interface

6.4.1.2 Usage

Python was used as the main programming language for the LSA implementation due to being extensible till the Ram size and having many third party packages for haonendling SVD calculations

6.4.2 DIVISI2

6.4.2.1 Description

Divisi is a sparse SVD toolkit for Python that is particularly designed for working with semantic networks.

6.4.2.2 Usage

Divisi was used to calculate SVD in a sparse way as the term vs. document matrix is so sparse so sparse SVD is more efficient than normal one

6.4.3 PyTables

6.4.3.1 Description

PyTables is a package for managing hierarchical datasets and designedto efficiently and easily cope with extremely large amounts of data. PyTables is built on top of theHDF5 library, using thePythonlanguage and theNumPypackage. It features an object-oriented interface that, combined with C extensions for the performance-critical parts of the code (generated usingCython), makes it a fast, yet extremely easy to use tool for interactively browse, process and search very large amounts of data. One important feature of PyTables is that it optimizes memory and disk resources so that data takes much less space (specially if on-flight compression is used) than other solutions such as relational or object oriented databases.

6.4.3.2 Usage

PyTables was used to save the term vs. document matrix in a h5 file format which caches part of the array in the Ram so as to be able to deal with array sizes bigger than that of the Ram. Also, the array is saved in a compressed format.

6.4.4 Numpy

6.4.4.1 Description

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

6.4.4.2 Usage

Numpy was used to perform matrix operations like: calculating the cosine similarity, calculate the vector representation of the document.

Appendix A

Appendix Title Here

A.1 Crawlers XML Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>

<config charset="UTF-8">

    <!-- set initial page -->
    <var-def name="home">http://ara.reuters.com/</var-def>

    <!-- define script functions and variables -->
    <script><![CDATA[
        /* checks if specified URL is valid for download */
        boolean isValidUrl(String url) {
            String urlSmall = url.toLowerCase();
            return (urlSmall.startsWith("http://ara.reuters.com/")&& !urlSmall.
    contains(" ")&& !urlSmall.contains("javascript"));
        }

        /* create filename based on specified URL */
        String makeFilename(String url) {
          String temp = url.replaceAll("http://|https://|file://", "");
          temp = temp.replaceAll("\\s","");
          temp = temp.replaceAll("\\:","");
            return temp.replaceAll("\\?","");
        }
    String makeFile(String url) {
    String temp = "http://ara.reuters.com";
    if(url!= null&&!url.toLowerCase().contains("javascript")&&url.startsWith("http:
    //ara.reuters.com/")){
      temp = url.replaceAll("http://|https://|file://", "");
      temp = temp.replaceAll("\\:","");
```

```
        temp = temp.replaceAll("\\s","");
    }
        return temp;
    }
    /* set of unvisited URLs */
    Set unvisited = new HashSet();
    unvisited.add(home);

    /* pushes to web-harvest context initial set of unvisited pages */
    SetContextVar("unvisitedVar", unvisited);

    /* set of visited URLs */
    Set visited = new HashSet();
]]></script>

<!-- loop while there are any unvisited links -->
<while condition="${unvisitedVar.toList().size() != 0}">
    <loop item="currUrl">
        <list><var name="unvisitedVar"/></list>
        <body>
            <empty>
                <var-def name="content">
                    <html-to-xml>
                        <http url="${currUrl}"/>
                    </html-to-xml>
                </var-def>

                <script><![CDATA[
                    currentFullUrl = sys.fullUrl(home, currUrl);
                ]]></script>

                <!--  saves downloaded page -->
                <file action="write" path="spider/${makeFilename(currentFullUrl
)}.html">
                    <var name="content"/>
                </file>


                <!-- adds current URL to the list of visited -->
                <script><![CDATA[
                    visited.add(sys.fullUrl(home, currUrl));
                    Set newLinks = new HashSet();
                    print(currUrl);
                ]]></script>

                <!-- loop through all collected links on the downloaded page --
>
                <loop item="currLink">
```

```
                        <list>
75                          <xpath expression="//a/@href">
                                <var name="content"/>
77                          </xpath>
                        </list>
79                      <body>
                            <script><![CDATA[
81                              String fullLink = sys.fullUrl(home, currLink);
                                if ( isValidUrl(fullLink.toString()) && !visited.
    contains(fullLink) && !unvisitedVar.toList().contains(fullLink) ) {
83                                  newLinks.add(fullLink);
                                }
85                           ]]></script>

87                          <file action="write" path="spiders/reuters/reuters${
    makeFilename(sys.fullUrl(home, currLink))}.txt" charset="UTF-8">
                            <xquery>
89                            <xq-param name="doc">
                              <html-to-xml>
91                             <try>
                          <body>
93                                      <http url="${makeFile(sys.fullUrl(home,
    currLink))}"/>
                                        </body>
95                          <catch>
                                No report file!
97                          </catch>
                      </try>
99                              </html-to-xml>
                            </xq-param>
101                           <xq-expression><![CDATA[
                              declare variable $doc as node() external;
103                             let $title := data($doc//title)
                                let $text := data($doc//div[@id="resizeableText"])
105                                 return
                                        <article>
107                                         <title>{data($title)}</title>
                                <text>{data($text)}</text>
109                                         </article>
                             ]]></xq-expression>
111                     </xquery>

113                     <![CDATA[ </reuters/reuters> ]]>
                        </file>
115                       </body>
                    </loop>
117             </empty>
            </body>
```

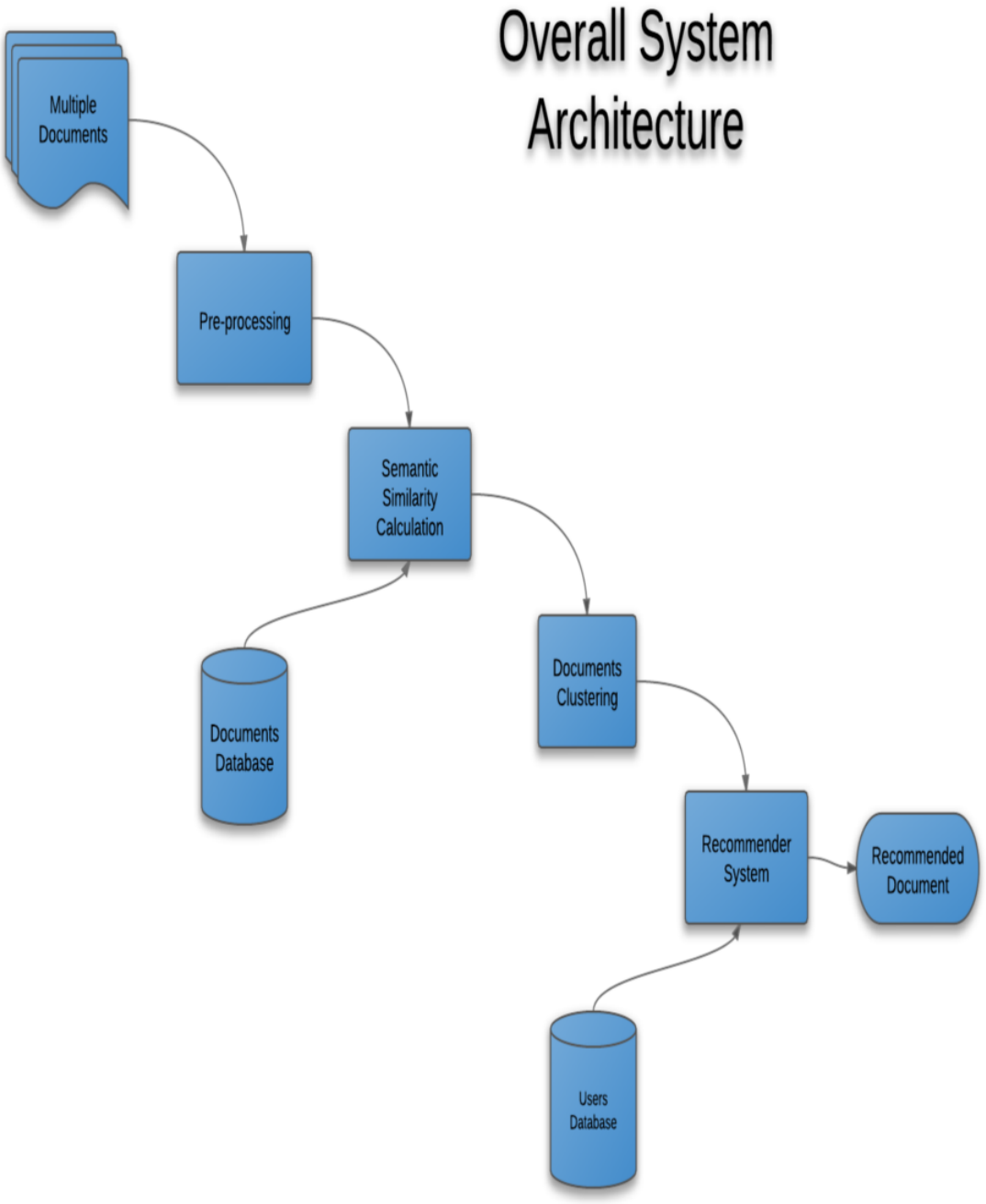
```
119         </loop>

121         <!-- unvisited link are now all the collected new links from downloaded
        pages  -->
            <script><![CDATA[
123             SetContextVar("unvisitedVar", newLinks);
            ]]></script>
125     </while>

127 </config>
```

A.2 JIGSAW Disambiguation

Architecture.png Architecture.png



Bibliography

[1] A. S. Arnold, J. S. Wilson, and M. G. Boshier. A simple extended-cavity diode laser. *Review of Scientific Instruments*, 69(3):1236–1239, March 1998. URL http://link.aip.org/link/?RSI/69/1236/1.

[2] Carl E. Wieman and Leo Hollberg. Using diode lasers for atomic physics. *Review of Scientific Instruments*, 62(1):1–20, January 1991. URL http://link.aip.org/link/?RSI/62/1/1.

[3] C. J. Hawthorn, K. P. Weber, and R. E. Scholten. Littrow configuration tunable external cavity diode laser with fixed direction output beam. *Review of Scientific Instruments*, 72(12): 4477–4479, December 2001. URL http://link.aip.org/link/?RSI/72/4477/1.