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

List of Figures

List of Tables

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 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.

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

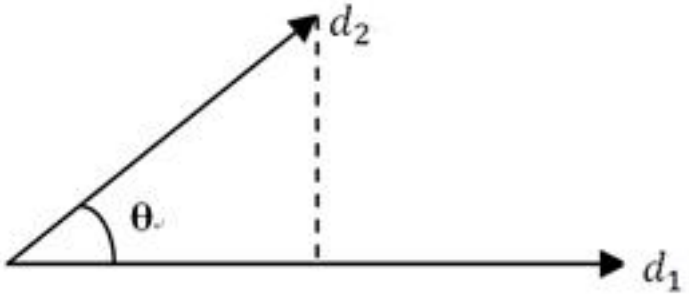

FIGURE 3.1: Angle Between Documents

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.

3.2.2 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

3.2.3 Weigthing Factor

3.2.3.1 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.

3.2.3.2 Latent Semantic Analysis Specificity:

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.

3.2.3.3 Inverse Document Frequency Specificity:

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

```
Input: Admissible labels Lwi = {lwi |t = 1..Nwi },i = 1..N
Output: Sequence of labels L = {lwi |i = 1..N }, with label lwi corresponding to word wi from the input
Build graph G of label dependencies
1: for i = 1 to N do
2: for j = i + 1 to N do
3: if j     i > M axDist then
4: break
5:end if
6:for t = 1 to Nwi do
7:for s = 1 to Nwj do
8:weight     Dependency(lwi , lwj , wi , wj )
9:if weight > 0 then
10:AddEdge(G, lwi , lwj , weight)
11:end if
12:             end for
13:         end for
14:     end for
15: end for
Score vertices in G
1: for all Va     V ertices(G) do
2: Score(Va )     Centrality(Va )
3: end for
Label assignment
1: for i = 1 to N do word.
2: lwi     argmax{W P (lwi )|t = 1..Nwi }
```

```
3: end for
```

3.2.5 ay7aga

Chapter 4

Chapter 4

Tools

4.1 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.

Chapter 5

Chapter 5

Recommendation System

5.1 Introduction

Recommender systems became an important research area since the appearance of the first papers on collaborative filtering since the mid-1990s [45, 86, 97]. 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 [61], movies by MovieLens [67], and news at VERSIFI Technologies (formerly AdaptiveInfo.com) [14]. Moreover, some of the vendors have incorporated recommendation capabilities into their commerce servers [78].

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 S R , where R is a totally ordered set (e.g., non-negative integers or real numbers within a certain range). Then for each user cC , we want to choose such item s'S that maximizes the users utility. More formally cC, sc'=argmaxuc,ss S (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 S space, but only on some subset of it. This means u needs to be extrapolated to the whole space C 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 1, where ratings are specified on the scale of 1 to 5. The symbol for some of the ratings in Table 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.

	K-PAX	Life of Brian	Memento	Not
Alice	4	3	2	
Bob	Ø	4	5	
Cindy	2	2	4	
David	3	Ø	5	

Table 1. A fragment of a rating matrix for a movie recommender sy

FIGURE 5.1: Angle Between Documents

Extrapolations from known to unknown ratings are usually done by:

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

b) 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 (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

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 ,si) assigned by user c to items siS 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 [7,89] and information filtering [10] 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 [8], which recommends Web pages to users, represents Web page content with the 100 most important words. Similarly, the Syskill Webert system [77] represents documents with the 128 most informative words. The "importance" (or "informativeness") of word ki in document dj is determined with some weighting measure wij 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 fi,j is the number of times keyword ki appears in document dj. Then TFi,j , the term frequency (or normalized frequency) of keyword ki in document dj, is defined as TFi,j=fi,jmaxzfz,j (2) Where the maximum is computed over the frequencies fz,j of all keywords kz that appear in the document dj. 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 (IDFi) is often used in combination with simple term frequency (TF i, j). The inverse document frequency for keyword ki is usually defined as IDFi=logNni (3) Then the TF-IDF weight for keyword ki in document dj is defined as wi,j=TFi,jIDFi (4) and the content of document dj is defined as Content(dj) = (w1j, wkj). As stated earlier, content-based systems recommend items similar to those that a user liked in the past [56, 69, 77]. 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 (wc1,,wck), where each weight wci denotes the importance of keyword ki 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 [85], can be used to compute ContentBasedProfile(c) as an "average" vector from an individual content vectors [8, 56]. On the other hand, [77] use a Bayesian classifier in order to estimate the probability that a document is liked. The Winnow algorithm [62] has also been shown to work well for this purpose, especially in the situations where there are many possible features [76]. In content-based systems, the utility function u(c, s) is usually defined as: uc, s= scoreContentBasedProfilec,Contents (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 Wc

and WS 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 Wc and WS , such as cosine similarity measure [7, 89]: uc,s=cosWc , WS =Wc . WS ——Wc——2 ——WS——2 =i=1kwi,cwi,si=1kwi,c2i=1kwi,s2 (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 wc , will represent such terms ki with high weights wic. 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 ws 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 [70, 77] and various machine learning techniques, including clustering, decision trees, and artificial neural networks [77]. 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, [77] use the nave Bayesian classifier [31] to classify unrated Web pages. More specifically, the nave Bayesian classifier is used to estimate the following probability that page pj belongs to a certain class Ci (e.g., relevant or irrelevant) given the set of keywords k1, j , , kn,j on that page: PCiK1,jKn,j (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 [101, 112], 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 [84, 111], which focuses on determining the extent to which documents should match a given query in order to be relevant to the user.

5.3 Limitation

5.3.1 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 [97]. 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 [97].

5.3.2 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 [98]. 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 [13], 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, [112] 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.

5.3.3 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 cjC 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=aggrc'C rc',s (9) 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: arc,s= 1Nc'Crc',s brc,s= kc'Csim(c,c')rc',s crc,s=rc+ kc'Csim(c,c')(rc',s- rc')

where multiplier k serves as a normalizing factor and is usually selected as k= 1c'C—sim(c,c')— and , and where the average rating of user c, rc in (10c) is defined as rc=1—sc—sScrc,s where Sc=sS rc,s (11) 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 Sxy be the set of all items co-rated by both users x and y, i.e. sxy=sS—rx,s ry,s . 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 Sx and Sy. However, some methods, such as the graph-theoretic approach to collaborative filtering [4], can determine the nearest neighbors of x without computing Sxy 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]: simx,y=sSxyrx,s-rxry,s-rysSxy(rx,s-rx)2sSxy(ry,s-ry)2 (12)

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=—Sxy —. Then, the similarity between two vectors can be measured by computing the cosine of the angle between them: sim(x,y)=cosx , y =x . y ——x——2 ——y——2 =i=1krx,sry,ssSxyrx,c2sSxyry,s2 (13) where x . y denotes the dot-product between the vectors x and 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.

5.4.3 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].

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.

5.4.4 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.

5.4.5 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.

5.4.6 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:

FEE TABLE HENA

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 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 di is a word. Each document then has a vector w, where element wi is the weight of word di for that document. If the document does not contain di then wi = 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: rw,m= w.m 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 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

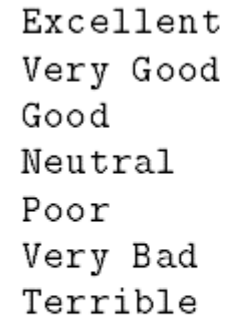

Figure 1: Ordinal scale users use to rank documents.

FIGURE 5.2: Angle Between Documents

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

Appendix A

Appendix Title Here

Write your Appendix content here.

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.