Bachelor Thesis Economics & Informatics

Ontology-Based News Items Recommendation in Athena

Author:
Frank Goossen
frankgoossen@student.eur.nl
306151fg

Supervisor:
Flavius Fransincar
frasincar@ese.eur.nl

Co-reader:
Frederik Hogenboom
fhogenboom@ese.eur.nl

Econometric Institute
Erasmus School of Economics
Erasmus University Rotterdam
PO Box 1738, NL-3000
Rotterdam, the Netherlands
July 15, 2009

Abstract

Recommending news items is traditionally done by term-based algorithms like TF-IDF. This paper concentrates on the benefit of recommending news items using a semantic algorithm (based on a domain ontology) instead of using a term-based algorithm. For this purpose, this paper proposes Athena, which is an extension to the Hermes framework. Athena employs a user profile to store concepts or terms found in news items browsed by the user. Based on this information, the framework uses a traditional method, TF-IDF and several ontology-based methods to recommend new articles to the user. The paper concludes with an extensive evaluation of each of the different methods, which shows that the use of an ontology significantly improves the performance of a recommender.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The last decade, the Web has become increasingly important in delivering news to the people. Many people like to read news articles and the Web is the best platform to find them. However, these news items are not personalized for the user interests. In this paper a new technique in delivering the most interesting news items to the user is proposed. Content-based news recommender systems

A common feature of news sites is a section with related news items, which are directly related (e.g., the same subject, but published some time ago) to the current news item. These related items are a recommendation to the user in the sense that if the user finds the current news item interesting, he might also be interested in the directly related items.

Recommending news items can be done by calculating the similarity between the current news item and the other news items. Traditionally, this similarity is calculated by an algorithm that is term-based, which practically means every word in a news item is taken into account. However, a news item often contains key concepts that capture the context of the article. By using the key concepts found in the news articles recommendations might be faster and more accurate than the word(term)-based recommenders.

The Hermes framework [1] is built on this method of relating news items. It uses an ontology to store these concepts and their relations. This paper focuses on a new way of recommending, based on concepts in the news item, employing some of the functionalities offered by Hermes.

In order to recommend news items first the user's browsing behavior is modeled. By recording a history of read news items, a profile of the user can be made. Based on this profile, it is possible to propose new news items that the user might find interesting.

This paper proposes Athena, which is an extension of the Hermes framework. Athena is able to observe user behavior and generate recommendations based on this behavior. The program uses several mathematical algorithms (recommenders) to compare 'new' news items (i.e. the user has

not read them yet) with the user profile. The news items having the highest similarity with the user profile are recommended to the user.

Section 1.1 identifies the main goals considered in this thesis. Then, in section 1.2, the methodology used in this thesis will be discussed. Finally, the overview of the structure of the thesis will be given in section 1.3.

1.1 Research Goals

The goal of this research is to investigate the benefit of recommending news items by using domain ontology-based recommenders with respect to traditional term-based recommenders, in this case TF-IDF. For this goal the following research questions can be created:

1. How to recommend items based on an ontology?
2. How do the ontology-based recommender algorithms perform with respect to classic recommender algorithms as TF-IDF?

1.2 Methodology

To answer the research question, the following research objectives are defined.

1. Creating an environment (Athena) which can be implemented in the Hermes News Portal [1]. Athena must provide all functionality a recommender algorithm needs
2. Creating ontology-based recommenders, which can be implemented in Athena. This will be discussed in section 4
3. Perform a statistical test, provided by the Student T-Test. Athena includes a testing environment, where test data can be processed. Another environment is used for test participants, which can rate news items.

1.3 Structure

The structure of this paper is as follows. First the related work is discussed, after that the paper continues with the Hermes framework and the Hermes News Portal (HNP), the implementation of the Hermes framework. Then the paper continues with the Athena framework and the implementation of Athena as a plug-in in the HNP. Finally, the proposed recommending methods are evaluated and the paper is concluded.

Chapter 2

Related Work

This section will describe the related work which has been studied for this research. Section 2.1 will discuss the existing news recommendation systems. Section 2.2 discusses the classic term-weighting method TF-IDF. Section 2.3 discusses another way of recommending, which will be used in Athena for recommending news items.

2.1 Existing News Recommendation Systems

Recommending news items or other documents based on the user's interest has attracted the interest of many researchers. Several adaptive Web-based news services have been developed which focus on personal recommendation of news items. These systems vary in application domain, platform, development methodology, levels of adaptivity, etc. Each system will be summarized shortly, and thereafter matched with the proposed semantic approach. Especially the TF-IDF approach of YourNews [2] will be extensively discussed, because the focus of the paper lies on investigating the benefit of ontology-based recommending with respect to classic recommending with TF-IDF.

2.1.1 Content-based news recommender systems

In content-based approaches, articles are recommended according to a comparison between their contents and the user profiles. The user profiles contain information about the users' content-based preferences. Both of these components have data-structures which are created using features extracted from the texts. A weighting scheme is often used to assign high weights to the most discriminating features/preferences, and low weights to the less informative ones.

In MyPlanet [3] ontologies are employed to benefit from semantics in recommending news items. In contrast to Athena, the ontologies are not

based on the standard ontology language OWL. Additionally the classification process of the news articles is performed by using heuristics, while Athena benefits from the advanced Natural Language Processing techniques employed by Hermes, including tokenization, part-of-speech tagging, word sense disambiguation, etc.

News Dude [4] is a personal news recommending agent that uses TF-IDF in combination with the Nearest Neighbor algorithm and considers the full text instead of focusing on key concepts and their relations by using ontologies. Additionally, while we use one user profile, News Dude first considers the short-term interests to look for similar items and if it fails, the long-term interests are considered.

The next related work is Daily Learner [5]. This is an adaptive news service which allows a user to first choose categories he or she wants to receive news about. Based on this user profile, the system delivers those stories that best match the users interests. A new article is matched with the user profile with the help of TF-IDF vectors and cosine similarity. Then, the user explicitly provides feedback using four rates (interesting, not interesting, more information, already known). Short-term interests are determined by analyzing the N most recently rated stories, based on the Nearest Neighbor Algorithm. Long-term interests are modeled with the Naive Bayes Classifier. Athena looks only at user's browsing history, and takes that as user profile. This history can be long or short. Every time a user reads a new news item, this item is registered in the user profile. The similarity is calculated by semantical algorithms, with the help of a knowledge base. Athena also implements the TF-IDF algorithm for evaluation of the semantical algorithms.

A final example of a news based recommendation system is YourNews [2]. It is a personalized news system, which intends to increase the transparency of adapted news delivery. It allows the user to view and edit his interest profile. To support this, YourNews highlights the key terms in news items.

The news items are represented in weighted vectors of terms. The weight of each term is calculated using TF-IDF [6]. Before creating those vectors, the text is filtered from stop words and each word is reduced to its stem using a Krovetz Stemmer [7]. The user profile is represented as a weighted vector of terms extracted from the user's view history. To compute the similarity between the user profile and the news articles the authors used the cosine similarity measure.

The differences between the approach of YourNews and the semantic approach is threefold. First [8] states, supported by Singhal's findings [9], that the performance of TF-IDF decreases as the length of the article, and the number of words, increases. In Athena the contents of an article is captured by the ontology, which means that each concept from the article that exists in the ontology is considered in the recommendation process. Thus instead of considering each individual word we only take in account the words that are known in the domain ontology. To create a recommendation method

based on a domain ontology we have made use of the Hermes framework [1]. Hermes provides us with a Semantic Web-based framework which includes a domain ontology for knowledge representation and natural language processing techniques for semantic text analysis. Second, the user profile we create is stored as a set of concepts instead of a set of keywords as employed by YourNews. For each concept we store the articles the user has read that contained the corresponding concept. Third because concepts are part of a domain ontology, there is more knowledge available about the context of the news item. Based on this information, one can find the concepts related to the ones in the news items. For instance an article that contains the concept *Apple* is known to be about a company and for that company it is known that the competitor is *Microsoft*. Athena makes use of such relation. YourNews focuses on personalizing news, with the help of TF-IDF and cosine vector similarity. Athena focuses on the benefit of using semantic methods to personalize news.

2.2 TF-IDF

The well-known term weighting method TF-IDF (term frequency-inverse document frequency) [6] has been used in most of the related work above. A classic approach in comparing documents is the use of TF-IDF together with the cosine similarity measure. TF-IDF is a statistical method used to determine the relative importance of a word within a document in a collection (or corpus) of documents.

Before calculating the TF-IDF values, the stop words are being filtered from the document. After stop word removal, the remaining words are stemmed using the Krovetz Stemmer [7] [10]. This process reduces words like 'process', 'processor', 'processing' and 'processed' back to their root word 'process'.

The TF-IDF measure can be determined by first calculating the term frequency (TF), which indicates the importance of a term t_i within a document d_j. By computing the inverse document frequency (IDF), the general importance of the term in a set of documents can be captured.

The objective is to compare any new document against the user profile. Therefore a vector is calculated for the user profile. This vector contains the TF-IDF value for 100 words with the highest TF-IDF value from the documents that have been read by the user. Subsequently in the same manner a vector, based on the total set of documents, is created for the new document that is being compared to the user profile. By calculating the cosine measure of the news item and the user profile, the similarity can be determined. The articles with the highest similarity value are considered to be the most similar to the user profile and are recommended to the user.

2.3 Semantic Relatedness

Apart from the traditional recommendation method, TF-IDF, also other other ways in comparing documents have been explored. [11] provides a practical approach to identify the relationships between RSS news items and measure the relatedness or similarity between the items. Their approach is based on the semantic relatedness between RSS [12] items. As in Athena they determine the relationship between words, using WordNet [13]. Their focus is on the semantic neighborhood of a word, in which general relationships as synonymy, hyponymy, meronymy, between words are considered. The difference with Athena is that we make use of an ontology. Besides the semantic relationship between words, the ontology covers relations like 'is-competitor-of', 'has-product', etc. Despite this difference, their method is applicable in our context, and therefore we will compare both approaches.

2.4 Concluding Remarks

This section described the related work studied for this research. Multiple existing recommender systems are introduced and briefly discussed. TF-IDF is also discussed, and a new method of recommending which is implemented in Athena. The related work described is important for this paper for it describes the area of this study: news items recommendation based on ontologies.

Chapter 3

Hermes: an Ontology Based News Service

Athena is an extension to the Hermes framework [1]. The HNP is an implementation of the Hermes framework, and Athena has been implemented as a plug-in for the HNP. This chapter briefly presents the Hermes framework and its implementation.

3.1 Hermes Framework

The Hermes framework is a set of methods necessary to build and maintain a news personalization service. The system can be described by input, internal processing, and output. The input is composed of predefined RSS feeds of news items and concepts selected by the user. The throughput is the classification of these news items, finding concepts from a knowledge base, and the updating of the knowledge base underlying the system, based on information discovered in news items. The output is defined as the personalized news items based on selected concepts.

The Hermes framework provides a semantic-based approach for retrieving news items related, directly or indirectly, to the concepts of interests from a domain ontology. Hermes stores it's knowledge base, the list of news items, and the relations between these news items in separate ontologies.

The knowledge base describes the general domain the user is interested in. It is used for several functionalities: the classification of (new) news items and the graphical representation used for selecting concepts of interest, the ontology graph. The knowledge base maintains itself using discovered information in news items.

The classification of news items is done after loading news from RSS feeds. Hermes uses an advanced Natural Language Processing (NLP) engine that uses techniques like, tokenization, part-of-speech tagging, word sense disambiguation, gazetteering, etc. This prepares the news item for querying.

During the classification, the system searches through the news item to find all knowledge base concepts. The news item is then stored together with the found concepts in a news ontology, so that future queries can be done in a fast and direct way, without going through the NLP steps again. All concepts of interest are stored in the domain ontology (knowledge base) and are represented by classes and individuals.

News querying is done by expressing the topics of interest using concepts from the domain ontology. The user can select these concepts with the help of the ontology graph. The user has the additional possibility to express constraints that the timestamps belonging to news items need to satisfy. The news querying step consists of two parts: query formulation, i.e., supporting the user to build queries, which is done in the ontology graph, and query execution, i.e., computing the results of query evaluation.

The results returned from the query are presented in the order of their relevance for the user query. For this purpose, for each returned news item a relevance degree is computed based on all the hits between the news item and the query concepts.

3.2 Hermes News portal

The Hermes News Portal (HNP) is the implementation of the Hermes framework we use. It allows the user to query the news and view the knowledge base. The domain ontology is represented in OWL [14], querying is done with SPARQL [15], and time functionalities were added to SPARQL, which results in tSPARQL [1]. Classification is done using GATE [16] and the WordNet [13] semantic lexicon.

The programming language Java [17] has been chosen since many libraries for manipulating, reasoning with, querying, and visualizing ontologies are available. Also, it combines well with GATE, which we use for most of the Natural Language Processing, since both GATE and its components and plug-ins are programmed in Java. Jena [18] is used for manipulating and reasoning with ontologies. It allows users to specify queries for the concepts of interest and temporal constraints, and retrieve the corresponding news items.

The user can do a search query with the help of the graphical tab of the HNP which is shown in Figure 3.1. This is the graphical interpretation of the knowledge base.

When the user commits its search query, the HNP translates this query into a SPARQL query. This SPARQL query will return only those news items which are most related to the users' concepts of interest. Finally, the HNP will present the most relevant news items to the user.

The HNP includes the possibility to add plug-ins, an example of an existing plug-in is a tab which allows loading news items from an RSS feed

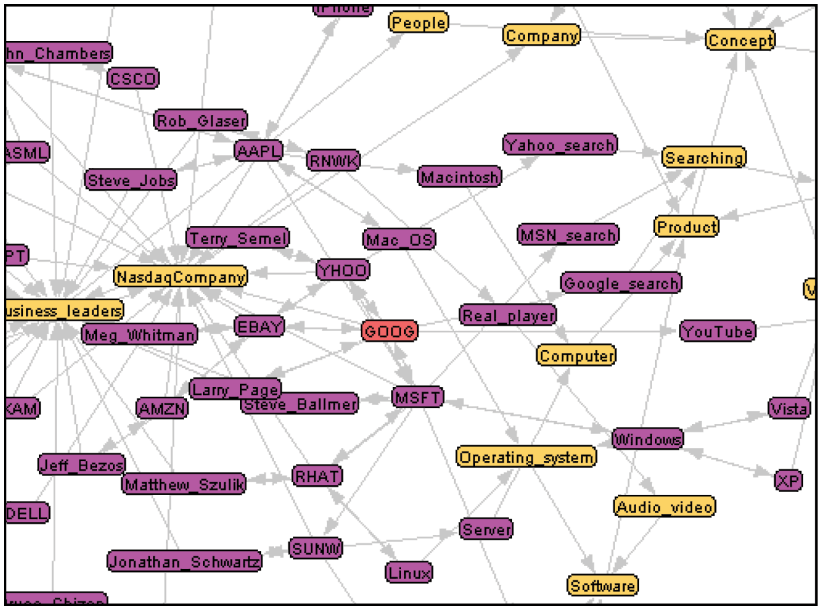

Figure 3.1: Example Graph in the Hermes News Portal

into the news ontology. A plug-in can be built as a new tab into the HNP.

3.3 Concluding

This chapter describeb the Hermes framework. The Hermes framework is a set of methods necessary to build and maintain a news personalization service. Its implementation is the Hermes News Portal. Both include the possibility to be extended, in this case with Athena, which is built as an extension to the Hermes framework, and implemented as plug-in in the Hermes News Portal.

Chapter 4

Athena: The Framework

Athena features a new method of news article recommendation based on the user's interest. At the heart of the method lies a domain ontology with the important concepts and relations from the corresponding domain. Athena is an extension to the Hermes framework. The method consists of three steps. First the user profile is constructed based on the articles the user has read. Second, the user profile and the articles need to be represented in a uniform way, and for this purpose the vector space model [19] is used. Finally, the similarity between the user profile and a new article is computed.

4.1 User Profile Construction

Recommending news items starts with building a user profile. Building a user profile can be defined as keeping track of which articles the user has read so far. Those articles will provide us with information about the user's interests. The user profile is constructed in different ways. For concept equivalence, binary cosine and Jaccard, the profile is a set of concepts from the articles the user has read. The semantic relatedness approach creates a vector with the distinct concepts from the user profile and assigns a weight to each concept. The ranked recommendation method also uses a vector of distinct concepts from the read articles and assigns a rank to each concept. The difference in user profile construction between the latter two approaches, is the method used to compute the corresponding weights.

4.2 Semantic Recommendation

In traditional forms of text comparison, all words in the text are considered. In addition to this there is no relation known between different words. For instance it is not possible to determine the relation between *Google* and *Microsoft*. But a user who is interested in news regarding to his stocks in Google might also be interested in news about *Microsoft*, because it is a

competitor of *Google*. Using an ontology that covers those relations might therefore be useful in recommending new articles. To illustrate how this is accomplished, a few related methods will be discussed, and concluded with a combination of methods.

4.3 Concept Equivalence

We start with a very simple technique which only considers the equivalent concepts. The ontology contains a set of n concepts:

$$C = \{c_1, c_2, c_3, \cdots, c_n\} \ . \tag{4.1}$$

The user profile consists of p concepts. This can be represented as the following set:

$$U = \left\{c_1^u, c_2^u, c_3^u, \cdots, c_p^u\right\}, \text{where } c_i^u \in C \ . \tag{4.2}$$

A new article can also be formulated as a set with q representing the number of concepts that appear in the article:

$$A = \left\{c_1^a, c_2^a, c_3^a, \cdots, c_q^a\right\}, \text{where } c_j^a \in C \ . \tag{4.3}$$

The interestingness of a new article is determined by looking at the intersection between the previous two sets:

$$\text{Similarity}(U, A) = \begin{cases} 1 & \text{if } |U \cap A| > 0 \\ 0 & \text{otherwise} \end{cases} \ . \tag{4.4}$$

If this results in 1, the article is considered interesting, otherwise it is considered not interesting.

4.4 Binary Cosine

Using sets of concepts makes it impossible to compute the similarity using the regular cosine measure. This measure requires a vector of values, like TF-IDF values. To compute the similarity between two texts we use the binary cosine similarity coefficient:

$$B(U, A) = \frac{|U \cap A|}{|U| \times |A|} \ , \tag{4.5}$$

where $|U \cap A|$ represents the number of items in the intersection of the elements from U and the elements from A. $|U|$ and $|A|$ are respectively the number of items in U and A.

4.5 Jaccard

The Jaccard similarity coefficient can be computed in a similar manner:

$$J(U, A) = \frac{|U \cap A|}{|U \cup A|} , \tag{4.6}$$

where $|U \cup A|$ is the union of the elements from U and A. Jaccard thus looks at the number of concepts from the article that appear in the user profile. Then it divides this by the number concepts that appear in either the profile or the article.

4.6 Semantic Relatedness

In [11] the focus is on the semantic relationship between words. The semantic neighborhood of a concept $c_i \in C$ is defined as the set of concepts related to the it via the synonymy ($\equiv$), hyponymy ($\prec$), and meronymy ($<<$). Our ontology covers more relations than only the linguistic relations. Therefore the semantic neighborhood of concept c_i includes each concept that is directly related to the concept c_i:

$$N(c_i) = \left\{ c_1^i, c_2^i, \cdots, c_n^i \right\} . \tag{4.7}$$

A text t_i can be described by a set of concepts:

$$CS_i = \left\{ c_1^i, c_2^i, \cdots, c_m^i \right\} . \tag{4.8}$$

When comparing two texts, t_i and t_j a vector in n-dimensional space can be created, according to the vector space model:

$$V_k = [\left\langle c_1^k, w_1^k \right\rangle, \cdots, \left\langle c_p^k, w_p^k \right\rangle] , \tag{4.9}$$

where $k \in \{i, j\}$ and w_i represents the weight associated to the concept c_i and $p = |CS_i \cup CS_j|$ is the number of distinct concepts in CS_i and CS_j. If the concept c_i is referenced in CS_i then $w_i = 1$ otherwise it is computed based on the maximum enclosure similarity it has with another concept c_j in its corresponding vector V_j. This takes into account the global semantic neighborhood of each concept as follows:

$$w_i = \begin{cases} 1 & \text{if freq}(c_i \text{ in } CS_j) > 0 \\ \max_j(\text{EnclosureSim}(c_i, c_j)) & \text{otherwise} \end{cases} . \tag{4.10}$$

$$\text{EnclosureSim}(c_i, c_j) = \frac{|N(c_i) \cap N(c_j)|}{|N(c_i)|} . \tag{4.11}$$

Finally the similarity between t_1 and t_2 is computed using the cosine measure:

$$\text{SemRel}(t_i, t_j) = \cos(V_i, V_j) = \frac{V_i \cdot V_j}{||V_i|| \times ||V_j||} \in [0, 1] \,, \tag{4.12}$$

where the nominator is the dot product of both vectors and the denominator is the multiplication of the magnitude of both vectors.

The advantage of this approach above concept equivalence, binary cosine and Jaccard, is that it also takes into account the related concepts of a concept that occurs in the text.

4.7 Ranked Recommendation

[20] describes an intuitive approach in working with adaptive media. For instance when you read something about concept c_1 which is related to concept c_2 and concept c_3 you increase not only your knowledge in c_1 but also in the other two.

Even though it is used in a different research field (adaptive hypermedia), the main idea can be applied in the ranked recommendation method. Each concept gets assigned a value, this value we call the rank. For example, when a user reads about Google, he might also be interested in its competitors, like Yahoo!, but also in news about its CEO Eric Schmidt. Both are considered to be direct relations of the concept Google. Therefore we increase the rank for Google, Yahoo! and Eric Schmidt. Unrelated concepts also need to be addressed. An unrelated concept is a concept that is not directly connected to the current concept. By decreasing the rank for such a concept we make the user profile adaptive to the user's main interest.

We define the set of related keywords to concept c_i as:

$$r(c_i) = \left\{ c_1^i, c_2^i, \cdots c_k^i \right\} \,. \tag{4.13}$$

R is described as the union of all related concepts to the concepts in the user profile:

$$R = \bigcup_{u_i \in U} r(u_i) \,. \tag{4.14}$$

And finally U_R is defined as the set of all concepts and corresponding related concepts, this is called the extended user profile:

$$U_R = U \cup R \,. \tag{4.15}$$

To calculate the final ranks for each concept we organize the concepts in a matrix. This is done because we have to assign a rank to each concept in the extended user profile for each concept the user has read about. Reading

Table 4.1: Rank matrix

	e_1	e_2	...	e_q
u_1	r_{11}	r_{12}	...	r_{11}
u_2	r_{21}	r_{22}	...	r_{2q}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
u_m	r_{m1}	r_{m2}	...	r_{mq}

about concept c_1 increases its value with 1.0. If concept c_2 is directly related to concept c_1, then its value is increased with 0.5. If there exists a concept, concept c_3, in the extended profile which is neither equal to concept c_1 nor is it related to concept c_1, its value is decreased with 0.1. These values are determined by experimenting with different values. Doing this for each concept results in a matrix with rank values. Summing the rows of the matrix results in a vector with the final ranks for each concept in the extended user profile. The columns contain the items from the extended user profile (U_R) and the rows contain the items from the user profile (U). Table 4.1 shows a rank matrix, where $e_i \in U_R$ and $u_i \in U$.

The user might have read one or more articles about a concept. The number of articles the user has read about concept u_i is called the weight w_i,

$$W = \{w1, w2, \cdots w_m\} \ . \tag{4.16}$$

Now we can calculate the value for each cell. This is done as follows:

$$r_{i,j} = w_i \times \begin{cases} +1.0 & \text{if } e_j = u_i \\ +0.5 & \text{if } e_j \neq u_i, e_j \in r(u_i) \\ -0.1 & \text{otherwise} \end{cases} \ . \tag{4.17}$$

The final rank for each concept can be computed by taking the sum of each column in the matrix:

$$\text{Rank}(e_j) = \sum_{i=1}^{m} r_{ij} \ . \tag{4.18}$$

Those sums are stored in a vector V_U. Each concept in the extended user profile now has a rank. Before we can compare the user profile with a new article, we need to ensure that the range of the ranks is [0,1]. This is because unranked concepts, which are concepts that may appear in the user profile but not in the new article, will get assigned a 0 to exclude them. This is done as follows:

$$V_U[v_i] = \frac{v_i - \min(v_u)}{\max(v_u) - \min(v_u)}, \text{where} \quad v_i \in V_U, v_u \in V_U \ . \tag{4.19}$$

Table 4.2: Rank matrix

	Yahoo!	Obama	China	Google	Apple	USA
Yahoo!	4	-0.4	-0.4	2	2	-0.4
Obama	-0.3	3	-0.3	-0.3	-0.3	1.5
China	-0.2	-0.2	2	-0.2	-0.2	1
Rank	3.5	2.4	1.3	1.5	1.5	2.1

With this we can compare the user profile to a new article that needs to be classified. Before we compute the vector for the new article we will clarify this method with a running example. For this example, the user profile is as follows:

$$U = \{\text{Yahoo!}, \text{Obama}, \text{China}\} \ .$$

The weights W for the corresponding concepts are

$$W = \{4, 3, 2\} \ ,$$

which means that the user in this example has read four articles that contained the concept 'Yahoo!' and three articles with 'Obama' and two articles with 'China' in the content. The sets of related concepts for each concept in the profile are as follows:

$$r(\text{Yahoo!}) = \{\text{Google}, \text{Apple}\} \ ,$$

$$r(\text{Obama}) = \{\text{USA}\} \ ,$$

$$r(\text{China}) = \{\text{USA}\} \ .$$

This creates a total set of related concepts R, which we define as

$$R = r(\text{Yahoo!}) \cup r(\text{Obama}) \cup r(\text{China}) = \{\text{Google}, \text{Apple}, \text{USA}\} \ .$$

The extended user profile is:

$$U_R = \{\text{Yahoo!}, \text{Obama}, \text{China}, \text{Google}, \text{Apple}, \text{USA}\} \ .$$

Table 4.2 shows the resulting rank matrix. Finally, the range of the ranks need to be adjusted to [0,1]. This results in vector V_U:

$$V_U = (1, 0.5, 0, 0.091, 0.091, 0.364) \ .$$

Before we continue with the example, we have to compute the vector with ranks for a new article that has not been read yet. The new article consists of a set of concepts, specified as A:

$$A = \{a_1, a_2, \cdots a_t\} \ . \tag{4.20}$$

For this article we define a vector containing the ranks. This vector is defined as V_A:

$$V_A = (s_1, s_2, \cdots s_t) \ , \tag{4.21}$$

$$s_i(e_i) = \begin{cases} \text{Rank}(e_i) & \text{if } e_i \in A \\ 0 & \text{if } e_i \notin A \end{cases} \ . \tag{4.22}$$

Each concept from the extended user profile that appears in the article is assigned the same rank as the one in V_U. The remaining concepts are assigned zero. Concepts appearing in the article but not in the profile are ignored. To compare the article with the user profile we propose to compute the extent to which the article fits the profile by dividing the sum of the ranks of concepts in the article by the sum of the ranks of the concepts in the user profile.

$$\text{Similarity}(V_A, V_U) = \frac{\sum_{v_a \in V_A} v_a}{\sum_{v_u \in V_U} v_u} \ . \tag{4.23}$$

The article with the highest similarity measure fits best with the user profile. To give some more insight in the comparison of the new article with the user profile we will continue the example.

The article that is being examined consists of three concepts and can be represented as:

$$A = \{\text{Google}, \text{USA}, \text{Vitamins}\} \ .$$

Then we can generate the vector that fits this article:

$$V_A = (0, 0, 0, 0.091, 0, 0.364) \ .$$

With V_U and V_A we can compute the extent to which the article fits the profile:

$$\text{Similarity} = \frac{0.091 + 0.364}{1 + 0.5 + 0 + 0.091 + 0.091 + 0.364} = 0.222 \ .$$

This approach is based on an intuitive method of modeling user interests. By simply increasing the value if a concept is equal or related and decreasing it when it is not directly related, the computation of the similarity becomes transparent. Additionally, by not only considering the concepts but also the related concepts, the user's interest is modeled such that it better fits the reality. The downside of the method lies in the fact that it needs a knowledge base that covers a large part of the domain. The quality of the knowledge base impacts the performance of this method.

4.8 Concluding Remarks

This chapter described Athena. Athena is a framework which can be implemented in the Hermes Framework. It provides news recommender algorithms which communicate with the knowledge base of Hermes. A recommender makes use of a user profile. A user profile consists of news articles the user has read, and their corresponding concepts. The chapter described the 5 ontology-based recommenders, how they are built, and how they are related to each other.

Chapter 5

Athena: The Implementation

As Athena is an extension to the Hermes framework, it has been implemented as a plug-in in the existing implementation of the Hermes framework, the Hermes News Portal (HNP). The implementation of Athena is done in the same language as the HNP, Java.

The user interface of Athena consists of 3 tabs: a browser for all news items, a tab for the recommendations, and a tab for evaluation purposes. The browser contains a number of news items sorted by date. This allow the user to browse freely through the news items, instead of browsing through query results as in the Hermes News Portal. Each item is presented with a title, an image which is related to the news item, and the date published. Section 5.1 describes how the user profile is constructed. The recommendation tab is discussed in the section 5.2, the last tab is used for evaluation in chapter 6.

5.1 User Profile Construction

The user profile is created from the articles the user has read. We define reading an article as, opening it into the Web browser.

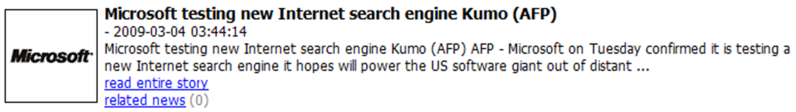

Figure 5.1: Query Result in Hermes News Portal

The Hermes News Portal [1] already enables the user to browse through query results. Figure 5.1 shows a query result in HNP. Clicking the 'read entire story' link will open the corresponding article in the Web browser. At this point Athena registers the article as read.

So far it was only possible to read news articles by first creating a search query and then browse through the results. To make it more user friendly

we created a tab that provides the user with all unread news articles sorted by publishing date in descending order. This enables the user to browse through all recent news items. Double-clicking the news item opens the article in the browser. Additionally, the article is being added to the user profile. Figure 5.2 shows a snapshot of the browsing process.

News articles are stored in a separate ontology, which is directly available from the Hermes framework. In an ontology each individual has a unique identifier, i.e. a Uniform Resource Identifier (URI). In the Hermes News Portal, this identifier is a hexadecimal representation of the title and publishing date of an article. This unique identifier enables us to store a minimal amount of information that identifies the read articles. Storing only the URI makes it possible to lookup any information about the article, i.e., the title, date and content. Besides a minimal need in storage capacity, the user profile also increases the flexibility of the system, because each recommender might need different information. An example is the TF-IDF recommender, which needs the actual content of the article. Therefore the only interesting elements in the user profile for the TF-IDF recommender are the URIs of the read news items. As explained in chapter 4, for the concept equivalence, binary cosine and Jaccard recommenders, the profile is a set of concepts from the articles the user has read. The semantic relatedness approach creates a vector with the distinct concepts from the user profile and assigns a weight to each concept. The ranked recommendation method also uses a vector of distinct concepts from the read articles and assigns a rank to each concept. The most important part of the user profile are the concepts. A concept is found in the Hermes knowledge base. Each news article contains zero or more concepts. The user's interest can therefore be determined from the visited news items. Therefore the user profile consists of concepts and corresponding articles. In Snippet 1 an example of a user profile is shown.

5.2 Semantic Recommendation

After reading several articles, the user can browse for the recommendations tab in Athena. In this tab the user can choose a type of recommender, and get recommended articles based on the user profile. Only one recommender can be chosen at a time. By clicking the refresh button, the recommender starts analyzing the user profile. After a short period of time, the recommender presents a list of news items that the user may find interesting. This list consists of the news items that the recommender ranked highest. Each news item is presented with a rank. The user can browse through the results, and by double-clicking at a news item, the procedure of registering information to the user profile is repeated, after that the user's web browser shows the concerning news article.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY up "http://www.hermes.com/up.owl#" >
    <!ENTITY kb "http://www.hermes.com/kb.owl#" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY news "http://www.hermes.com/news.owl#11" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://www.hermes.com/up.owl#"
     xml:base="http://www.hermes.com/up.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:news="http://www.hermes.com/news.owl#11"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:kb="http://www.hermes.com/kb.owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:up="http://www.hermes.com/up.owl#">
    <owl:Ontology rdf:about=""/>

    <owl:ObjectProperty rdf:about="#foundIn">
        <rdfs:domain rdf:resource="#Concept"/>
        <rdfs:range rdf:resource="#NewsItem"/>
    </owl:ObjectProperty>

    <owl:Class rdf:about="#Concept">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="#NewsItem">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="&owl;Thing"/>

    <Concept rdf:about="&kb;China">
        <foundIn rdf:resource="&news;fa3b5f68"/>
    </Concept>

    <Concept rdf:about="&kb;Company">
        <foundIn rdf:resource="&news;fb1dad00"/>
        <foundIn rdf:resource="&news;ff93d1d90"/>
    </Concept>

    <Concept rdf:about="&kb;IPhone">
        <foundIn rdf:resource="&news;fb1dad00"/>
    </Concept>

    <Concept rdf:about="&kb;Networking">
        <foundIn rdf:resource="&news;fa3b5f68"/>
        <foundIn rdf:resource="&news;ff93d1d90"/>
    </Concept>

    <Concept rdf:about="&kb;Risk">
        <foundIn rdf:resource="&news;fb1dad00"/>
    </Concept>

    <NewsItem rdf:about="&news;fa3b5f68"/>
    <NewsItem rdf:about="&news;fb1dad00"/>
    <NewsItem rdf:about="&news;ff93d1d90"/>
</rdf:RDF>
```

Snippet 1: Example of a User Profile

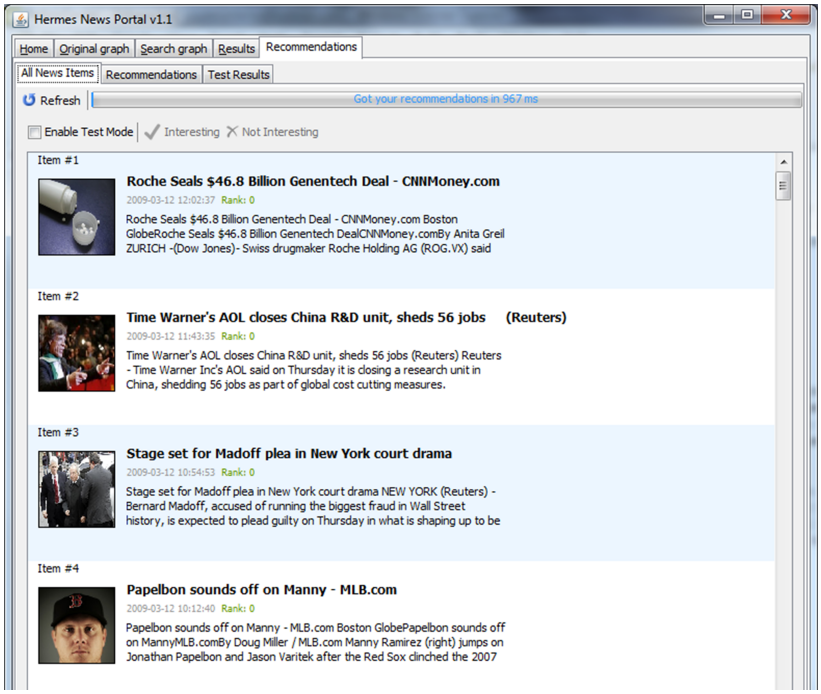

Figure 5.2: Browsing All News Items

In this tab, shown in Figure 5.3, we have included a concept cloud (based on a tag cloud), which lists all the concepts which have been stored in the user profile. When a concept is read in multiple articles, the font size gets larger. Finally we have included a feature which highlights the concepts and related concepts found in the article in different colors.

Additionally, Athena provides a testing environment (see figure 5.4) for evaluation purposes. It is possible to load and save test data, to recall test results. Important values for testing like the cut-off value or the number of iterations can be changed. This tab will be used in chapter 6

5.3 Concluding Remarks

This chapter described the implementation of the Athena framework. The implementation of the framework consists of building a new tab into the Hermes News Portal. This tab consists of 3 different tabs, where the user can browse, get recommendations, or test the recommending algorithms. The chapter also described how the user profile is built, and how it is used by the recommenders.

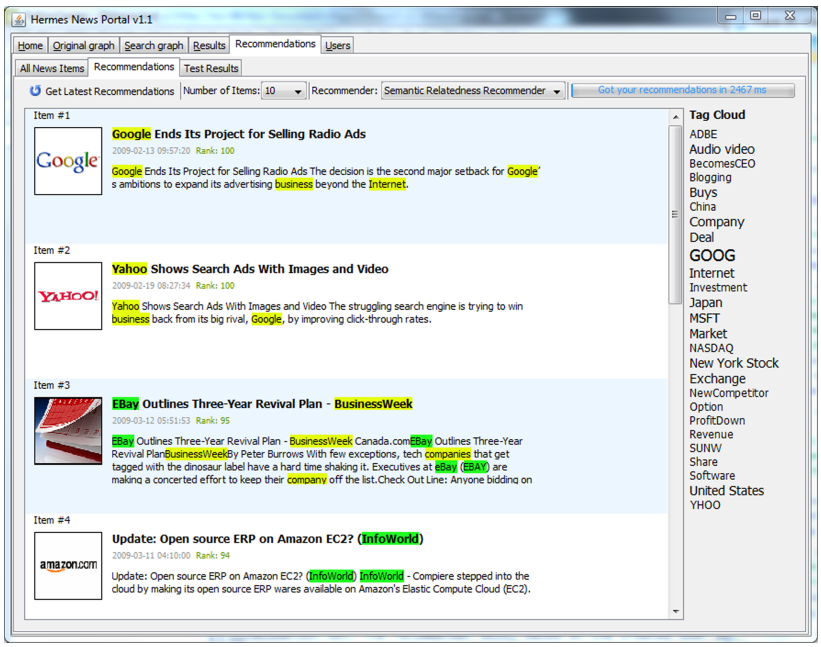

Figure 5.3: Recommendation Tab

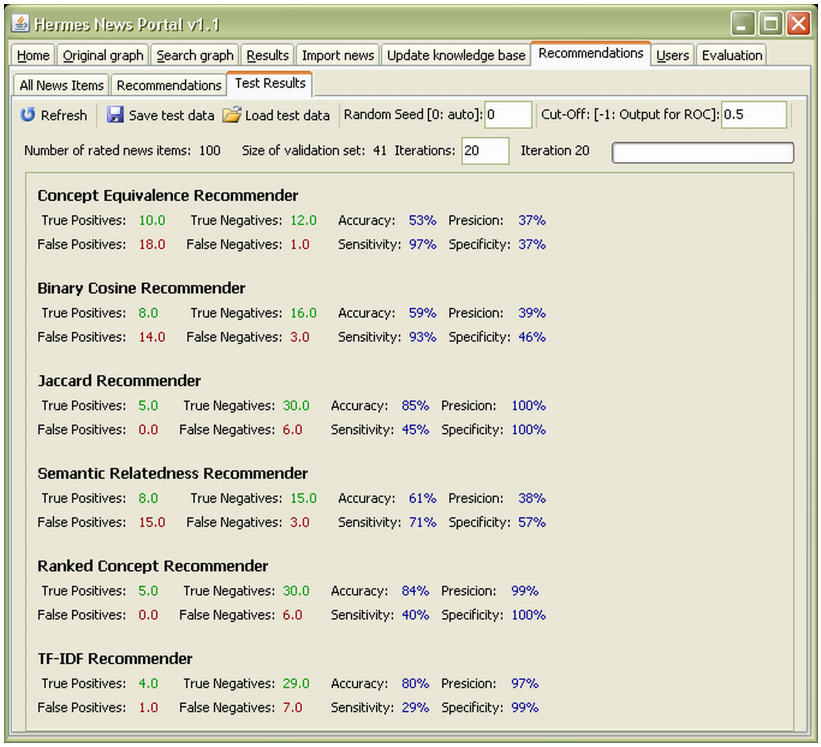

Figure 5.4: Evaluation Tab

Chapter 6

Evaluation

The research goal is to investigate whether ontology-based recommenders perform better than a classic recommender based on TF-IDF. To evaluate the ontology-based recommenders, a test method and test environment have been built. Also a classification method from the business intelligence field is used. The results are evaluated by hypothesis testing and ROC curves. The resulting tables and figures referenced are listed in Appendix A to maintain a clear structure.

6.1 Experimental Setup

For the test environment, a website has been created where the news items are displayed one at a time. For each article the user has to indicate whether it is interesting or not. The experiment requires the user to keep a clear profile in mind, which is *all articles that are related to Microsoft, and its products and competitors*. After the user finishes the test, a resulting set of articles is created, consisting of the articles and their rating. Athena then processes the result set.

The processing of the result set is based on supervised learning. The result set will be randomly split into two different sets, the training set (60%) and the validation set (40%). The two sets will be filled with a relatively equal number of items rated as interesting. The training set is used to create a user profile. Each item that is marked as interesting will be added to this profile. The validation set is used by each recommender to determine for each news item the similarity with the user profile. An article is considered to be interesting if the similarity is higher than a predefined cut-off value, in this case 0.5, otherwise it is classified as not interesting.

To determine the performance of a recommender, several measures are calculated using a confusion matrix, as showed in table 6.1.
The confusion matrix can be used to compute several statistical measures. First the *accuracy*, the percentage of the predictions that are correct.

Table 6.1: Confusion Matrix

		Actual Value	
		Interesting	Not Interesting
Predicted Value	Interesting	TP	FP
	Not Interesting	FN	TN

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6.1}$$

Then the *precision*, which is the percentage of positive predictions that are correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.2}$$

Followed by the *sensitivity* or *recall*, the percentage of positive labeled instances that were predicted as positive.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.3}$$

Finally, the *specificity* which is defined as the percentage of negative labeled instances that were predicted as negative.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{6.4}$$

With this method and those measures we are able to gather sophisticated information about the the different recommendation methods. For each method we can determine whether it performs good or bad from a certain point of view.

One run might give unreliable results, because of the random value for splitting the result set. Therefore, Athena performs the testing process for 20 iterations with different random values for splitting the result set, and calculates the average performance of the recommenders.

6.2 Experiment

The goal of the experiment is to investigate whether the ontology-based recommendation approaches perform statistically better than the TF-IDF approach. Also the ontology-based recommenders are compared to each other. This done with the help of hypothesis testing.

19 persons will participate in the experiment. From each person a resulting set of articles is created, and processed by Athena. When Athena finishes processing, the resulting performance measures are saved. This data can be used for statistical purposes.

This experiment will focus on hypothesis testing, and ROC curves. Each comparison will consist of 2 samples of different recommenders. Only the same performance measures will be compared. One sample consists of 19 values, each representing the average measured performance of a recommender on the result set of one user. In the test, the average of this sample will be taken as mean performance. A comparison will consist of 2 recommenders, respectively recommender A and recommender B. The comparison is created in such a way that the expected better performing recommender will be A, and the expected worse performing recommender will be B. In table 6.2 the comparison are listed.

$$\begin{aligned} &\mu_1\text{: Mean performance recommender A ,} \\ &\mu_2\text{: Mean performance recommender B .} \end{aligned} \tag{6.5}$$

With the performance defined as accuracy, precision, sensitivity (recall) or specificity.
With a level of significance of 95%, the corresponding hypotheses are as follows. First the two-sided hypothesis:

$$H_{0_a} : \mu_1 = \mu_2, H_{1_a} : \mu_1 \neq \mu_2, \text{with } \alpha = 0.05 \; . \tag{6.6}$$

And, if we can reject H_{0_a}, the one-sided hypothesis:

$$H_{0_b} : \mu_1 < \mu_2, H_{1_b} : \mu_1 > \mu_2, \text{with } \alpha = 0.05 \; . \tag{6.7}$$

First, all the ontology-based recommenders are compared with the TF-IDF recommender. Hereafter, the ontology-based recommenders are compared to each other. The comparisons are made in the following order:
Each comparison consists of a comparison on each of the four performance measures (see formulas 6.1, 6.2, 6.3 and 6.4). Because there are two related samples, the Student's T-Test can be used, which is explained by the following formulas.

$$t_{n-1} = \frac{\overline{d}}{s_d/\sqrt{n}} \tag{6.8}$$

$$\overline{d} = \frac{\sum_{i=1}^{n} d_i}{n} \tag{6.9}$$

$$s_d^2 = \frac{\sum_{i=1}^{n} d_i^2 - nd^2}{n-1} \tag{6.10}$$

6.3 Experimental Results: Student T-Test

After processing the results of the 19 participants, we are able to do the Student T-Tests. Table 6.3 shows a clear overview of all the average perfor-

Table 6.2: Tests

No.	μ_1	μ_2
1	Ranked Recommendation	TF-IDF
2	Jaccard	TF-IDF
3	Semantic Relatedness	TF-IDF
4	Binary Cosine	TF-IDF
5	Concept Equivalence	TF-IDF
6	Ranked Recommendation	Jaccard
7	Ranked Recommendation	Semantic Relatedness
8	Ranked Recommendation	Binary Cosine
9	Ranked Recommendation	Concept Equivalence
10	Jaccard	Semantic Relatedness
11	Jaccard	Binary Cosine
12	Jaccard	Concept Equivalence
13	Semantic Relatedness	Binary Cosine
14	Semantic Relatedness	Concept Equivalence
15	Binary Cosine	Concept Equivalence

mances of each recommender. Note that a recommender performing worse on average does not necessarily perform worse in a T-Test.

Table 6.3: Averages of each measure for all recommenders

Recommender	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**
Concept Equivalence	49.1%	32.2%	92.4%	34.4%
Binary Cosine	55.0%	34.7%	86.9%	43.6%
Jaccard	81.8%	80.9%	45.7%	94.6%
SemRel	59.3%	34.5%	69.6%	56.6%
Ranked	81.3%	78.8%	37.5%	95.7%
TF-IDF	79.9%	83.6%	30.3%	96.9%

First we have to determine the critical values. Since the α of the first (two-tailed) test is 0.05 and n = 19, the corresponding critical value for the first test is 2.101. The decision rule for the first test: *Reject* H_{0_a} *if* $-2.101 > t > 2.101$. For the second (one-tailed) test the corresponding critical value is 1.734 For the second test the decision rule is: *Reject* H_{0_b} *if* $t > 1.734$.

All the t-values calculated are listed in table A.1. Values which reject both H_{0_a} and H_{0_b} are printed bold, values which reject only H_{0_a} are printed italic. Note these italic printed values tell us that the recommender A actually performs worse than recommender B. Values which do not reject H_{0_a} are printed in a normal font.

From this table many interesting facts can be deduced. The accuracy and

the sensitivity of the Ranked Recommendation approach score significantly higher than the corresponding measurements of the TF-IDF approach. At the same time, the precision of the TF-IDF approach scores significantly higher than the precision of the Ranked Recommendation approach. This means that the TF-IDF recommender is better in *classifying items correctly as interesting*, or in other words: *the percentage of items classified as interesting by the recommender being correctly classified* is higher. However, the Ranked Recommendation approach scores significantly better in *classifying items correctly, both interesting and not interesting* and *classifying interesting rated items correctly as interesting* or in other words: *the percentage of items rated as interesting by the user being correctly classified* is higher.

The Jaccard recommender scores significantly different and better than the TF-IDF recommender on accuracy and sensitivity, but does not perform significantly different on precision. All the other ontology-based approaches (Concept Equivalence, Binary Cosine and Semantic Relatedness) perform significantly worse than TF-IDF on accuracy, precision and specificity, and significantly better than TF-IDF on sensitivity.

Regarding the comparisons between the ontology-based recommenders and the classic TF-IDF recommender we note that two of the recommenders, the Ranked Recommendation approach and the Jaccard approach, succeed in performing significantly better than TF-IDF on more than one performance measure. Both approaches score higher than TF-IDF in accuracy and sensitivity. All of the other ontology-based approaches score higher than TF-IDF on one measure, the sensitivity. Although the sensitivity is quite low for each recommender, the ontology-based recommenders succeed in performing better than the TF-IDF approach. The overall low performance on sensitivity can be explained by the fact that recommenders rather have no uninteresting items in their result list, than all the interesting items. This is caused by the cut-off value, used when rating an item interesting or not. Usually, a higher sensitivity causes a lower precision, which may explain why the precision is overall high with respect to sensitivity. One should ask himself what is more important: to have all the interesting articles in the resulting list (sensitivity), or to *not* have uninteresting articles in the resulting list (precision).

Tables A.2, A.3, A.4 and A.5 show whether H_{0_b} was rejected. The tables compare each recommender to all the other recommenders. *Yes* means H_{0_b} was rejected, *No* means H_{0_b} was accepted, and therefore it is proven that $\mu_1 < \mu_2$. The sign '-' means H_{0_a} was not rejected, and therefore $\mu_1 = \mu_2$. Tables A.6 and A.7 list all the rejections/acceptions.

6.4 Experimental Results: ROC Curves

Another evaluation method of the experiment is an ROC (receiver operating characteristic) curve. This is a graphical plot of the sensitivity versus fall-out (the percentage of uninteresting rated articles classified interesting by the recommender, 1 - specificity) as the discrimination threshold (the cut-off value described in section 6.1) is varied. Figure 6.4 shows the aggregated ROC curve for all the recommenders. Each line is representing one recommender. The ROC curve is based on one optimized result set (see section 6.1, to gain as reliable results as possible. The squares drawed on the lines point the sensitivity and fall-out values corresponding to the standard cutoff value of 0.5.

We can see that the TF-IDF recommender performs overall the highest. With the results of the Student T-Test in mind we know that the TF-IDF recommender outperforms most of the recommenders (only the Jaccard recommender is not significantly different) on precision. A high precision means that the TF-IDF recommender rather classifies interesting items as uninteresting, than uninteresting items as interesting. The TF-IDF recommender also performs high on specificity and low on sensitivity. This tells us more of the same: TF-IDF classifies a lot items as uninteresting, while some of these items are actually rated interesting. The fall-out of TF-IDF is quite low compared to other recommender. This can be explained by the same reason as above. The fall-out value is the percentage of uninteresting items that were erroneously classified as interesting. The TF-IDF recommender classifies most of the items as uninteresting, and the few items classified as interesting are correctly classified, which causes a high precision and low fall-out.

As the squares indicate (see also table A.9, the Jaccard recommender and the Ranked Recommendation approach perform better than the TF-IDF recommender with a corresponding cut-off value of 0.5. As showed in table A.8 and A.10 the TF-IDF approach performs best in the ROC curve when the cut-off value is very low (0.00 - 0.25). We can see that the TF-IDF approach performs quite low when the cut-off value gets higher than 0.5, compared to the Ranked Recommendation approach and the Jaccard approach. The low cut-off value causes a very high sensitivity, because of the large amount of interesting classified items. Though the Ranked Recommendation approach scores higher than TF-IDF in sensitivity at a corresponding cut-off value of 0.25, it also classifies some interesting items as uninteresting, causing a fall-out rate of 3.3% while TF-IDF keeps the fall-out at 0%. Again like with the Student T-Test, one should ask himself what is more important: to have all the interesting articles classified as interesting (sensitivity high), or to have no uninteresting items classified as interesting (precision high, fall-out low).

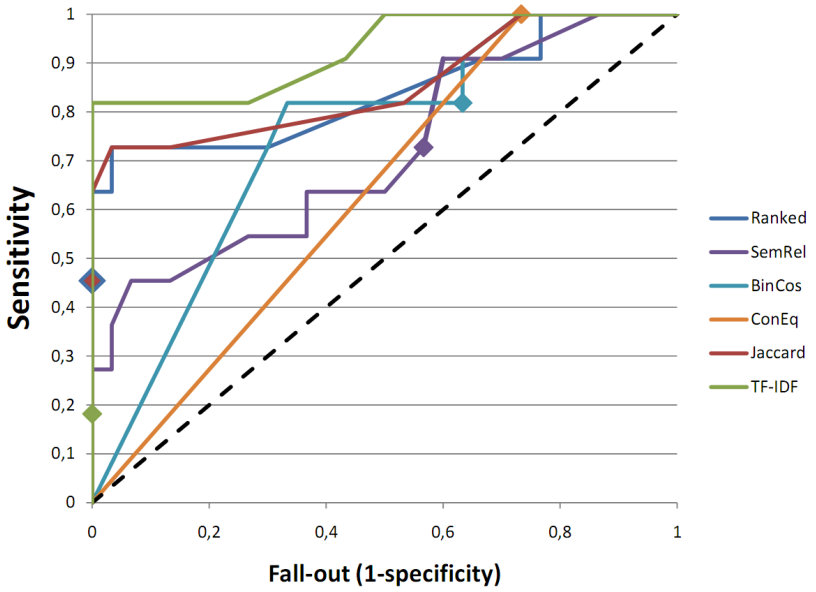

Figure 6.1: ROC Curve Aggregrated

6.5 Concluding Remarks

Overall the experiment results prove that the use of an ontology for recommending improves the accuracy and sensitivity of the results. The Student T-test shows that the Jaccard approach and the Ranked Recommendation approach score significantly higher than the TF-IDF in both performance measures.

The ROC curve shows us that the TF-IDF recommender has the overall best performance with regard to sensitivity versus fall-out. However, at the cut-off value of 0.5, which is standard in Athena, the ontology-based recommender Jaccard and the Ranked Recommendation approach both outperform TF-IDF.

The Jaccard algorithm scores overall quite high. This can be explained by the simplicity of the recommender. Instead of looking through relations of concepts (the Ranked Recommendation approach), it just counts the interesting concepts in a news article, and divides it by the sum of all the concepts in the article and the user profile.

The Ranked Recommendation approach scores lower than expected. Its accuracy and sensitivity score quite high, but its precision is significantly lower compared to the precision of the TF-IDF approach. This means that the Ranked Recommendation approach rather haves a lower amount of false negatives, which causes higher sensitivity, than having a lower amount of false positives, which causes a higher precision and higher specificity.

Chapter 7

Conclusions and Future Work

This paper describes Athena, an extension to the Hermes framework that provides several methods for news item recommendation based on the user's interests. Section 7.1 describes the conclusions of this paper, and section 7.2 discusses future work.

7.1 Conclusion

In Chapter 1, the following research questions were defined:

- How to recommend items based on an ontology?
- How do the ontology-based recommender algorithms perform with respect to classic recommender algorithms as TF-IDF?

7.1.1 How to recommend items based on an ontology?

At the heart of the system is the ontology provided by the Hermes framework. This ontology contains the concepts and the relations between the concepts. With these relations, more information about each concept is available than only the concept itself. This allows Athena to consider different articles interesting than existing technologies that employ TF-IDF. This is because the ontology-based appraoches do not only consider the concepts that appear in the article, but also the ones that are related to them. The ontology-based recommenders of Athena retreive a relatively high sensitivity, because they try to find as much interesting items as possible, while the TF-IDF recommender tries to get a minimal amount of uninteresting items in the recommendations. First the user profile needs to be constructed from the user's readings. With the concepts from the articles the user has read,

a user profile can be represented as a vector space model, which is also used to describe news articles.

Different methods that to employ ontologies in comparing the user profile with a new article are described. Concept equivalence is a simple, intuitive method that looks for articles that contain at least one of the concepts from the profile. This method does not take into account the number of concepts found in the news article. In order to take into account these concepts, we have used Binary Cosine and Jaccard. Those methods compute the similarity between the article and the profile. A more advanced method also takes into account the semantic relatedness between different concepts, which are provided by the underlying ontology. A weight is assigned to each concept based on the neighborhood and the enclosure similarity (based on the knowledge base). Additionally a fifth ontology-based method is described, which also uses the relationship between the concepts. It takes the concepts from the user profile and combines these with the related concepts to create the extended user profile. Each concept from the extended user profile and the news articles that need to be classified, gets a rank. With the rank vector of the extended user profile and the rank vector of a new article, the similarity between the user profile and a new article is computed. The difference with the semantic relatedness approach is twofold: the calculation of the weight or rank that is assigned to each concept and the calculation of the similarity between the user profile and the news article.

The traditional approach that uses TF-IDF considers the full text of the news articles. However, as [8] made a comparison with different lengths of documents, the performance decreases as documents get larger. In the ontology approach, we do not consider the full text, but only the concepts that exist in the knowledge base. With the semantic knowledge about the concepts it is possible to consider more than just the text at hand. The strength of the algorithm depends on the quality of the knowledge base.

7.1.2 How do the ontology-based recommender algorithms perform with respect to classic recommender algorithms as TF-IDF?

In the experiment a domain ontology was used that covered business news and politics. As the results show, two ontology-based recommenders, the Ranked Recommendation approach and the Jaccard approach, perform significantly better than the classic TF-IDF approach on two measures: accuracy and sensitivity. The Jaccard approach shows that its specificity and precision are not significantly different from TF-IDF approach. This means that it can be stated that the use of an ontology significantly may improve the performance of a recommender, depending on the the algorithm chosen. Though the Ranked Recommendation approach performs significantly better than the TF-IDF approach on accuracy and sensitivity, it performs sig-

nificantly lower on precision. This means that the Ranked Recommendation approach rather haves a lower amount of false negatives (articles classified uninteresting while they were rated by the user as interesting), which causes a higher sensitivity, than having a lower amount of false positives (articles classified uninteresting while they were rated by the user as interesting), which causes a higher precision. From the Student T-Test results can be stated that ontology-based recommenders are significantly better than TF-IDF on sensitivity. The high sensitivity is strongly related to the standard cut-off value of 0.5 in Athena. This is illustrated in the ROC curve (figure 6.4). This ROC curve shows that the TF-IDF recommender has the overall best performance with regard to sensitivity versus fall-out. However, at the cut-off value of 0.5, which is standard in Athena, the ontology-based recommender Jaccard and the Ranked Recommendation approach both outperform TF-IDF. Overall, the results show a clear difference between the ontology-based recommenders and TF-IDF. While ontology-based recommenders perform significantly higher than the TF-IDF approach in accuracy (Ranked Recommendation Approach and Jaccard Approach) and sensitivity (all ontology-based approaches), only the Jaccard approach manages to perform significantly indifferent on precision compared to the TF-IDF approach. This means that it can be stated that the use of ontology-based recommender algorithms has advantages with regard to accuracy and sensitivity, but can have disadvantages with regard to precision. As stated in chapter 6, one should ask himself what is more important: to have as many interesting items classified as interesting as possible (high sensitivity), or to have no uninteresting items classified as interesting (high precision).

7.2 Future Work

This paper has shown that using a domain ontology improves the accuracy and sensitivity of the recommendations to the user. The knowledge base that is used, is partly created by a domain expert and takes a lot of effort. Future research should consider to focus on automatically creating and maintaining such a knowledge base to support ontology based recommendation methods. When the knowledge base is extended, it may be interesting to test the Ranked Recommendation Approach, because this algorithm is strongly dependable on the size and depth of the knowledge base. Besides the improvement of the knowledge base, the algorithms can be improved as well. In the Ranked Recommendation Approach the focus was on a limited number of relations between concepts, for instance only the directly related concepts. However, concepts might be related to each other on different levels, i.e., concepts might not be directly related to each other but there might exist a relation with one or more concepts between them.

Appendix A

Tables

Table A.1: T-Values for all tests

Test	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**
Ranked vs TF-IDF	**3.084**	*-2.510*	**3.824**	-1.721
Jaccard vs TF-IDF	**2.156**	-1.392	**5.620**	-1.712
SemRel vs TF-IDF	*-14.915*	*-12.777*	**13.042**	*-39.593*
BinCos vs TF-IDF	*-17.816*	*-13.414*	**20.771**	*-23.271*
ConEq vs TF-IDF	*-21.412*	*-13.189*	**29.329**	*-30.630*
Ranked vs Jaccard	-0.667	-1.435	*-4.242*	1.137
Ranked vs SemRel	**17.535**	**12.266**	*-15.713*	**28.983**
Ranked vs BinCos	**20.418**	**12.291**	*-26.832*	**23.423**
Ranked vs ConEq	**25.101**	**12.506**	*-41.731*	**30.680**
Jaccard vs SemRel	**13.868**	**10.576**	*-11.542*	**19.000**
Jaccard vs BinCos	**23.598**	**10.623**	*-11.694*	**22.305**
Jaccard vs ConEq	**27.876**	**10.822**	*-16.090*	**29.110**
SemRel vs BinCos	**2.708**	-0.175	*-5.918*	**4.449**
SemRel vs ConEq	**7.493**	**3.458**	*-9.297*	**8.273**
BinCos vs ConEq	**12.866**	**5.265**	*-5.782*	**21.433**

Table A.2: Accuracy: reject H_{0_b} ($\mu_1 > \mu_2$) ?

		Recommender B					
		ConEq	**BinCos**	**Jaccard**	**SemRel**	**Ranked**	**TF-IDF**
Recomm. A	**ConEq**	-	No	No	No	No	No
	BinCos	Yes	-	No	No	No	No
	Jaccard	Yes	Yes	-	Yes	-	Yes
	SemRel	Yes	Yes	No	-	No	No
	Ranked	Yes	Yes	-	Yes	-	Yes
	TF-IDF	Yes	Yes	No	Yes	No	-

Table A.3: Precision: reject H_{0_b} ($\mu_1 > \mu_2$) ?

		Recommender B					
		ConEq	**BinCos**	**Jaccard**	**SemRel**	**Ranked**	**TF-IDF**
Recomm. A	**ConEq**	-	No	No	No	No	No
	BinCos	Yes	-	No	-	No	No
	Jaccard	Yes	Yes	-	Yes	-	-
	SemRel	Yes	-	No	-	No	No
	Ranked	Yes	Yes	-	Yes	-	No
	TF-IDF	Yes	Yes	-	Yes	Yes	-

Table A.4: Sensitivity: reject H_{0_b} ($\mu_1 > \mu_2$) ?

		Recommender B					
		ConEq	**BinCos**	**Jaccard**	**SemRel**	**Ranked**	**TF-IDF**
Recomm. A	**ConEq**	-	Yes	Yes	Yes	Yes	Yes
	BinCos	No	-	Yes	-	Yes	Yes
	Jaccard	No	No	-	No	Yes	Yes
	SemRel	No	-	Yes	-	Yes	Yes
	Ranked	No	No	No	No	-	Yes
	TF-IDF	No	No	No	No	No	-

Table A.5: Specificity: reject H_{0_b} ($\mu_1 > \mu_2$) ?

		Recommender B					
		BinCos	**BinCos**	**Jaccard**	**SemRel**	**Ranked**	**TF-IDF**
Recomm. A	**ConEq**	-	No	No	No	No	No
	BinCos	Yes	-	No	-	No	No
	Jaccard	Yes	Yes	-	Yes	-	-
	SemRel	Yes	-	No	-	No	No
	Ranked	Yes	Yes	-	Yes	-	-
	TF-IDF	Yes	Yes	-	Yes	-	-

Table A.6: Results for all tests

Test	**Accuracy**	**Precision**
Ranked vs TF-IDF	**Reject** H_{0_A}, **Reject** H_{0_b}	*Reject* H_{0_A}, *Accept* H_{0_b}
Jaccard vs TF-IDF	**Reject** H_{0_A}, **Reject** H_{0_b}	Accept H_{0_A}
SemRel vs TF-IDF	*Reject* H_{0_A}, *Accept* H_{0_b}	*Reject* H_{0_A}, *Accept* H_{0_b}
BinCos vs TF-IDF	*Reject* H_{0_A}, *Accept* H_{0_b}	*Reject* H_{0_A}, *Accept* H_{0_b}
ConEq vs TF-IDF	*Reject* H_{0_A}, *Accept* H_{0_b}	*Reject* H_{0_A}, *Accept* H_{0_b}
Ranked vs Jaccard	Accept H_{0_A}	Accept H_{0_A}
Ranked vs SemRel	**Reject** H_{0_A}, **Reject** H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
Ranked vs BinCos	**Reject** H_{0_A}, **Reject** H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
Ranked vs ConEq	**Reject** H_{0_A}, **Reject** H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
Jaccard vs SemRel	**Reject** H_{0_A}, **Reject** H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
Jaccard vs BinCos	**Reject** H_{0_A}, **Reject** H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
Jaccard vs ConEq	**Reject** H_{0_A}, **Reject** H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
SemRel vs BinCos	**Reject** H_{0_A}, **Reject** H_{0_b}	Accept H_{0_A}
SemRel vs ConEq	**Reject** H_{0_A}, **Reject** H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
BinCos vs ConEq	**Reject** H_{0_A}, **Reject** H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}

Table A.7: Results for all tests (continued)

Test	**Sensitivity**	**Specificity**
Ranked vs TF-IDF	**Reject** H_{0_A}, **Reject** H_{0_b}	Accept H_{0_A}
Jaccard vs TF-IDF	**Reject** H_{0_A}, **Reject** H_{0_b}	Accept H_{0_A}
SemRel vs TF-IDF	**Reject** H_{0_A}, **Reject** H_{0_b}	*Reject* H_{0_A}, *Accept* H_{0_b}
BinCos vs TF-IDF	**Reject** H_{0_A}, **Reject** H_{0_b}	*Reject* H_{0_A}, *Accept* H_{0_b}
ConEq vs TF-IDF	**Reject** H_{0_A}, **Reject** H_{0_b}	*Reject* H_{0_A}, *Accept* H_{0_b}
Ranked vs Jaccard	*Reject* H_{0_A}, *Accept* H_{0_b}	Accept H_{0_A}
Ranked vs SemRel	*Reject* H_{0_A}, *Accept* H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
Ranked vs BinCos	*Reject* H_{0_A}, *Accept* H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
Ranked vs ConEq	*Reject* H_{0_A}, *Accept* H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
Jaccard vs SemRel	*Reject* H_{0_A}, *Accept* H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
Jaccard vs BinCos	*Reject* H_{0_A}, *Accept* H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
Jaccard vs ConEq	*Reject* H_{0_A}, *Accept* H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
SemRel vs BinCos	*Reject* H_{0_A}, *Accept* H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
SemRel vs ConEq	*Reject* H_{0_A}, *Accept* H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}
BinCos vs ConEq	*Reject* H_{0_A}, *Accept* H_{0_b}	**Reject** H_{0_A}, **Reject** H_{0_b}

Table A.8: Performances with cut-off value (0.25)

Recommender	**Sensitivity (%)**	**Fall-out (%)**
Ranked	72.7	3.3
Jaccard	63.6	0.0
SemRel	91.0	70.0
BinCos	100	73.3
ConEq	100	73.3
TF-IDF	63.6	0.0

Table A.9: Performances with **standard** cut-off value of 0.5

Recommender	**Sensitivity (%)**	**Fall-out (%)**
Ranked	45.5	0.0
Jaccard	45.5	0.0
SemRel	72.7	56.7
BinCos	81.8	63.3
ConEq	100	73.3
TF-IDF	18.2	0.0

Table A.10: Performances with cut-off value (0.75)

Recommender	**Sensitivity (%)**	**Fall-out (%)**
Ranked	36.4	0.0
Jaccard	27.3	0.0
SemRel	45.5	6.7
BinCos	72.7	30.0
ConEq	100	73.3
TF-IDF	9.1	0.0

References

[1] Frasincar, F., Borsje, J., Levering, L.: A Semantic Web-Based Approach for Building Personalized News Services. International Journal of E-Business Research (IJEBR) **5**(3) (2009) 35–53

[2] Ahn, J., Brusilovsky, P., Grady, J., He, D., Syn, S.Y.: Open User Profiles for Adaptive News Systems: Help or Harm? In: 16th international conference on World Wide Web, ACM (2007) 11–20

[3] Kalfoglou, Y., Domingue, J., Motta, E., Vargas-Vera, M., Shum, S.B.: myPlanet: an ontology-driven Web-based personalised news service. In: International Joint Conferences on Artificial Intelligence 2001: Workshop on Ontologies and Information Sharing. (2001)

[4] Billsus, D., Pazzani, M.J.: A Personal News Agent that Talks, Learns and Explains. In: The Third Annual Conference on Autonomous Agents. (1999) 268–275

[5] Billsus, D., Pazzani, M.J.: User Modeling for Adaptive News Access. User Modeling and User-Adapted Interaction **10**(2) (2000) 147–180

[6] Salton, G., Buckley, C.: Term Weighting Approaches in Automatic Text Retrieval. Technical report, Cornell University (1987)

[7] Krovetz, R.: Viewing Morphology as an Inference Process. In: ACM Conference on Research and Development in Information Retrieval, ACM (1993) 191–202

[8] Bogers, T., van den Bosch, A.: Comparing and evaluating information retrieval algorithms for news recommendation. In Konstan, J.A., Riedl, J., Smyth, B., eds.: ACM Conference On Recommender Systems, ACM (2007) 141–144

[9] Singhal, A., Salton, G., Mitra, M., Buckley, C.: Document Length Normalization. Information Processing and Management **32**(5) (1996) 619–633

[10] Guzman-Lara, S.: KStem Java Implementation. University of Massachusetts Amherst (2007) from: http://ciir.cs.umass.edu/cgi-bin/downloads/downloads.cgi.

[11] Getahun, F., Tekli, J., Richard, C., Viviani, M., Yetongnon, K.: Relating RSS News/Items. In: 9th International Conference on Web Engineering. (2009)

[12] Winer, D.: RSS 2.0 Specification (2003)

[13] Fellbaum, C., ed.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge, MA (1998)

[14] Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneijder, P., Stein, L.A.: OWL Web Ontology Language Reference. Technical report, World Wide Web Consortium (W3C) (2004) from: http://www.w3.org/TR/owl-ref/.

[15] Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. Technical report, W3C (2005) from: http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050217/.

[16] Cunningham, H.: GATE, a General Architecture for Text Engineering. Computers and the Humanities **36** (2002) 223–254

[17] Gosling, J., Joy, B., Steele, G., Bracha, G.: The Java Language Specification, Third Edition. Addison-Wesley (2005) from: http://java.sun.com/docs/books/jls/download/langspec-3.0.pdf.

[18] McBride, B.: Jena: A Semantic Web Toolkit. IEEE Internet Computing **6**(6) (2002) 55–59

[19] Salton, G., Wong, A., Yang, C.S.: A Vector Space Model for Automatic Indexing. Communications of the ACM **18**(11) (1975) 613–620

[20] Bra, P.D., Aerts, A.T.M., Houben, G.J., Wu, H.: Making General-Purpose Adaptive Hypermedia Work. In: WebNet 2000 Conference, AACE (2000) 117–123