Master in Artificial Intelligence (UPC-URV-UB)

Master of Science Thesis

Intelligent RSS Tool

Markus Mettälä

Advisors: Prof. Javier Béjar, Universitat Politècnica de Catalunya
Prof. Aristides Gionis, Aalto University

August 19, 2013

Universitat Politècnica de Catalunya

Degree Programme in Artificial Intelligence

ABSTRACT OF
MASTER'S THESIS

Author: Markus Mettälä

Title:
Intelligent RSS Tool

Date: August 19, 2013 **Pages:** xi + 83

Supervisors: Prof. Javier Béjar, Universitat Politècnica de Catalunya
Prof. Aristides Gionis, Aalto University

Easy access to a wide range of information available online enables people to explore this information with an ambition to explore interesting content even more. This opportunity often leads to a problem of finding interesting and relevant information from the sea of knowledge. This problem is often referred to as the information overload problem, which is getting harder and harder to deal with as the amount of information available online grows. In this thesis, one source of information is exploited and organized in such a way that the task of discovering new content is made easier.

We use Really Simple Syndication (RSS) as our source of information and two methods to categorize it: document clustering with K-Means and Latent Dirichlet Allocation (LDA). We use the textual information that the RSS contains, each RSS feed usually contains a specific set of topics. Our first goal is to perform document clustering to the data, in order to generate meaningful clusters with the help of natural language processing (NLP) techniques to preprocess the data.

Our second goal is to analyze the clustered RSS feeds and exploit the similarities between the documents to generate meaningful user models based on user feed subscriptions. The third goal is to provide relevant recommendations based on the user models we have learned. We combine the current state-of-the-art methods and present novel methods to compare feeds. We exploit WordNet shallow ontologies in our novel method to create generalized representations of the feeds. The final goal is to develop a functional application that can leverage the methods we developed with the help of machine learning libraries. The method we propose is a combination of document clustering techniques, text similarity, feed modeling and recommendation system.

The results of our experiments show that K-Means clustered documents combined with recommendations based on the feed contents yield the best results. Using WordNet to measure the similarity of words provides also promising results. Further exploring the advantages of using semantic similarities would be an interesting research topic in the document similarity measures.

Keywords: RSS, document clustering, recommendation system, text classification, vector space distance measures, shallow ontologies

Language: English

Acknowledgements

This work was carried out in the Dual Degree Programme between Aalto University in Finland and Universitat Politcnica de Catalunya in Spain. I would like to express my gratitude to both my supervisors for all the most valuable advice, patience and comments they provided for me. Furthermore, I am grateful for all the people who have helped me throughout this thesis by providing their opinions or by sharing their knowledge with me.

Finally, I would like to thank my family, friends and fellow students for supporting, encouraging and motivating me throughout my studies.

Vantaa, August 19, 2013

Markus Mettälä

Abbreviations and Acronyms

CDbw	Compose Density Between and Within clusters
df	Document Frequency
HTML	Hypertext Markup Language
idf	Inverse Document Frequency
IUF	Inverse Update Frequency
LDA	Latent Dirichlet Allocation
NLP	Natural Language Processing
RSS	Really Simple Syndication
STC	Suffix-Tree Clustering
tf-idf	Term Frequency - Inverse Document Frequency
WordNet	Database of English words, their connections and syntactic categories
XML	Extensible Markup Language

Contents

Chapter 1

Introduction

Humans have usually a strong thirst for knowledge, which can be quenched in modern days quite easily by searching for information online. Some people are interested of particular topics and other people of their own particular topics, and there can exists groups of people who share their interests. These behavior patterns provide us a reason to try and quench the information thirst with the most appropriate methods in machine learning.

Partly due to the widespread access to internet, there exists vast amounts of information available online for everyone in many different formats. Typically the information on the internet tends to be updated frequently and distributed in many forms. One form of distribution is to use feeds which usually contain summarized information and a link to the full-text resource. The content of each feed varies and usually it resembles the information presented on that particular source of information.

The problem with this situation is that it is hard for a user to find new and interesting information from the available feeds. It is easy to subscribe to these feeds from a site which is frequently visited by the user, but for the user to find a new site with similar content is difficult. This is a common problem of finding new sources of information from the internet, commonly referred as the information overload problem.

Another problem is to filter out the information that is not interesting to the user. There can be a situation where the user is only interested of a subset of, for example, the news presented in a website or a feed. In this case the user has to browse through all the information and distinguish the most interesting news items. This is a problem of information filtering, to present the items that are most likely interesting to the user in an ordered list.

In this thesis we present one approach to solve these problems by using information retrieval techniques. In short, our approach builds a model of the user's interests based on his or her subscriptions and uses the textual

content of the feeds to recommend feeds that the user might find interesting.

We used natural language processing and document clustering techniques to form an understanding of the content in the user's feeds. Having structured an understanding of the data, we use approaches from the field of recommendation systems to find feeds that should be recommended to the user. We experimented with several different approaches and their combinations in an attempt to find the best way to solve the problem of finding the relevant feeds.

We used traditional methods, such as vector distance computations, and methods that we found not so frequently used in this domain of recommendation systems. The more infrequent methods include document clustering, comparisons of top-k lists, graph based approaches and a new approach that we present in this thesis, which is based on WordNet.

Based on our experiments, the best results can be achieved with K-Means document clustering and modeling the users based on the feeds they subscribe to. Furthermore, some methods such as using the feed content with Latent Dirichlet Allocation based document clustering for recommendations and incorporating WordNet information yields promising results.

The remaining sections of this thesis are organized as follows. In chapter 2, we begin with a more detailed explanation of the problem and present some recent approaches to solve it. We continue with background information in chapter 3, and in chapter 4 we present our approach in more detail. In chapter 5, we present the tools we used in our application and chapter 6 presents our experiments and the results. We evaluate the results in chapter 7 and finally we conclude this thesis in chapter 8, where we present discussion and draw conclusions.

Chapter 2

The RSS Recommendation Problem

We will begin this chapter by motivating the problem and then introduce the common approaches in recommendation systems for text feeds. Finally, we present the current applications and their approaches and describe how we plan to improve these approaches.

There are already some RSS aggregators which provide recommendations of feeds or items based on user preferences. However, there are not that many more recent approaches, thus providing us a reason to approach the problem with more recent methods developed in the field of recommendation systems and information retrieval.

2.1 Why Recommend Feeds

According to Built With [48], which is a trends, intelligence and internet research company, there are over 31.2 million websites providing information using feeds on their website, with over 9.2 million feeds provided using RSS. This means that 29.4% of websites that provide a feed, provides that feed using RSS, and from a total number of almost 81 million websites included in Built With's analysis 12.3% of them provide RSS feed.

When we consider the 9.2 million feeds provided on the web with RSS, it is very hard to find a feed that is interesting. Obviously, if the site one frequently visits provides feed in RSS, it is easy enough to subscribe to it. Still, the problem is how to find more feeds of the same subject without any tools. In addition, because of the vast selection of available feeds this can cause the decision to be even harder to make.

Edmunds and Morris discuss the problem of information overload in the

context of business organisations [14]. In the paper the authors present a literature review on information overload and provide many definitions for it; for example, information overload can mean that a person has more relevant information that can be understood or it can mean that there is a large amount of information, out of which some may be relevant. In our case we consider information overload as the latter case, where there is a large amount of feeds available and out of them some may be relevant.

Let us consider a user who is interested in, for example, technology and is familiar with a website or two and is subscribing their RSS feeds. According to data provided by Built With on their website, there are over 3000 RSS feeds in the domain of technology. How would the user be able to find the similar feeds from the available 3000 feeds. One option would be to manually go through them all and select the interesting ones, however, this might lead to bad decisions, which is a consequence of information overload, and also going through manually all the feeds is an irrational solution, thus to solve this problem is an interesting task. To solve the problem we can use the computational power of modern computers and some intelligent algorithms. Now that the problem we are faced with is more clear, we can proceed with the explanation of the data we are working with.

2.2 RSS - Really Simple Syndication

RSS is a syndication format for Web content and its abbreviation originates from Really Simple Syndication defined by the RSS advisory board [42]. RSS is similar to Extensible Markup Language (XML) in syntax and it is actually a dialect of XML.

RSS document contains elements which describe the content inside these elements, for example, RSS document starts with a `<rss>` element. Inside `<rss>` element there is a `<channel>` element which contains the actual content of the RSS feed. `<channel>` element contains metadata such as `<title>`, `<description>` and publication date, `<pubDate>`, for the feed. Inside the `<channel>` element there are one or more `<item>` elements, which contains the information of the items in the feed. Usually `<item>` contains at least a short description, or in some cases maybe even the full text of the item, and the publication date. For example, a simple RSS feed can be in the following format:

```
<rss>
   <channel>
      <title>Example RSS feed</title>
```

```
        <link>http://example.org/</link>
        <description>
           This example RSS feed shows the syntax for RSS
        </description>
        <item>
           <description>
              A sample RSS item
           </description>
           <pubDate>
              Tue, 16 July 2013 11:55:05 GMT
           </pubDate>
        </item>
        <item>
           <description>
              A sample RSS item 2
           </description>
           <pubDate>
              Mon, 15 July 2013 10:24:00 GMT
           </pubDate>
        </item>
     </channel>
  </rss>
```

RSS is really a simple format, as the name suggests, and easy to use and additionally the most widely spread syndication format according to Built With [48]. Furthermore, because RSS is a standard format it is really straightforward to create a tool for it, which can be used to read any RSS feed available. These features of RSS makes it a perfect choice for a source of data in this thesis.

2.3 Current state-of-the-art Applications

Current applications can be roughly divided to two categories: client-based and Web-based applications. The main difference between the categories is that client-based approaches enable the application to use more detailed information on the user preferences; for example, the application could inspect the contents of users' documents in the hard drive to obtain a more detailed user model. This information is not available in Web-based applications, but the advantage of using such an application is that the feeds are monitored constantly for new additions. In contrast, client-based applications do not

read the feed unless the user's computer is powered on and the application is running, therefore, these applications might overlook some interesting items in the feed which are no longer available in the feed when the user returns.

Some more academic oriented approaches have been developed for which the details are available; however, some commercial or other types of approaches do not provide the details. The next sections will describe these approaches in more detail when available.

2.4 Recent Study of the Problem

Ji and Zhou [26] present the most recent study on the field of recommendation systems and RSS feeds. The authors describe the current state of recommendation applications for RSS feeds at the time of the writing of the paper. According to the authors, these applications have some different approaches to the problem. Intersynd [38] recommends feeds from similar users which are defined by sharing at least 20% of the feeds. NectaRSS [43] and CRESDUP (Content recommendation system based on private dynamic user profile) [11] analyze the textual contents of the feeds and use this information to find similar feeds.

Ji and Zhou [26] compared in their paper different similarity metrics to discover similar feeds. The results presented in the paper indicate that the best performing metrics are the fraction of favourite feeds from all feeds subscribed to by the user as well as the inverse update frequency of the feed itself.

2.5 Intersynd

Intersynd is a middleware application that computes the recommendations for the application and does not try to replace the RSS reader [38]. Intersynd's recommendation algorithm is based on shared feeds between neighbouring users which are defined as sharing 20% feeds [38]. The recommended feeds are the most commonly appearing ones in the neighbourhood weighted by the inverse of feed list sizes.

Intersynd also incorporates diversity in the recommended feeds by adding some bias to feeds that are similar but not so common. Diversity enables the users to experience more different feeds that are still interesting [38].

2.6 NectaRSS

NectaRSS [43] uses an approach to build user models in an incremental approach. Each time the user uses NectaRSS, it is considered as a session and if there are previous sessions the model built is updated with the information gathered at the current session. The information used in NectaRSS is a vector space representation of the text information contained in the RSS feed.

NectaRSS does not use all the information available in the RSS feed, but instead it only relies on the headlines of the feed items. Their approach is based on the assumption that the information is newslike which allows the simplification that the headline contains all the distinctive information. The similarity is then computed by comparing the terms in the headlines that the user has read in previous sessions and the terms in the new headline.

The results presented in the paper by Samper et al. [43] show that their approach is significantly better than randomly choosing items to recommend. Furthermore, they conclude that the precision of the recommendations improve as more sessions are added to the users.

2.7 Google Reader

Perhaps one of the most developed and most widely known RSS aggregators is Google Reader [19]. As Google does not describe in detail their recommendation methods, we can draw some conclusions from the descriptions on their website.

"Your recommendations list is automatically generated - it takes into account your existing feed subscriptions. Aggregated across many users, this information can indicate which feeds are popular among people with similar interests. For instance, if a lot of people subscribe to feeds about both peanut butter and jelly, and you only subscribe to feeds about peanut butter, Reader will recommend that you try some jelly." [19]

From this description, we can conclude that their recommendation engine computes some sort of user model and locates the most similar user from which the feeds are collected. According to the description, it appears that Google Reader focuses on the content discovery retrieving unknown topics that the most similar users subscribe to rather than topics to which they are already subscribed.

Google announced in March 2013 that they will terminate the Google Reader service due to decreasing usage, and it was discontinued on July 1st, 2013 [20].

2.7.1 Consequences of Google Reader Shutdown

Even though, Google stated that the reason for shutting down Google Reader was the decline in usage, there has been an active conversation on the topic on the Web. For example, a simple query on Google Search about the discontinuation of Google Reader provides many blog results discussing the alternative solutions.

Also many similar feed aggregation services have mentioned on their blogs that the discontinuation notice from Google Reader has created a spike on their new users. For example, Feedly mentioned on their blog that during the two weeks that followed the Google discontinuation announcement, they received over 3 million new users [17]. In addition, NewsBlur mentioned on their blog that after Google's announcement their count of daily users rose from 1500 users to 50000 users [9].

According to these signals, we can assume that there is still a wide demand for Google Reader replacement, albeit this demand was turning to be too small for Google to maintain Google Reader.

2.8 NewsBlur

NewsBlur is one of the many services that received new users from Google Reader after its termination. NewsBlur is completely open source application licensed under MIT License. It also includes intelligence on the recommendations of the feeds based on the likes of users [8]. The approach they have chosen, to the best of our knowledge, uses Fisher's linear discriminant classifier to classify the feed as interesting or not.

2.9 How Our Approach Improves the Field

None of the approaches previously presented mention to perform any clustering on the feeds, to the best of our knowledge. We believe this is an important step to group the similar feeds using document clustering techniques. Not only can this improve the quality of the recommendations but also narrow down the search space for finding the feeds to recommend. Our approach also uses the similar methods presented in other approaches, we use the textual content of the feeds and the user preferences, in addition we also present a new approach to find similar feeds. We leverage the feed weighting approaches presented, and present a novel method to model the quality of the feed.

To emphasize, our approach does not try to contain all the features that, for example, NewsBlur or Google Reader provides, but instead our approach mostly expands the field of similarity measures used in feed recommendation systems. Nonetheless, our approach is not in any way specific to feed recommendation and the same methods can be applied to many fields of text document recommendations.

Chapter 3

Background

In this section we present the methods we used in this thesis. We present first the methods that are used to preprocess the data, and continue with the methods that rely on the results of the previous methods. We begin this chapter by giving a broad introduction to text mining, then we continue with document clustering. Finally, we describe similarity measures and finish this chapter with an introduction to recommendation systems.

3.1 Introduction

To motivate the reader to the topics of this chapter we begin with an explanation of our dataset. As previously mentioned, our application uses RSS feeds and consequently we will have textual data at our disposal. Even though, the data is initially RSS we do not consider it to be relevant to keep it in that format, we will remove all the RSS elements as described later in this section.

The dataset we used contained over 1200 feeds, which we used as described in the experiments section 6.1. Each feed contains items which we store as text, furthermore each file is stored to a folder which then forms the contents of the feed. From each item in the feed we collect the description and title to a text file, also we collect the publication times. We do not store the publication time in the files where the contents are stored to prevent effecting our text clustering algorithms. The resulting files represent the documents we cluster in our approach.

The length of the file contents, in practice the number of characters in a feed, varies between different feeds and we can use the length differences as a feature of a feed to model the quality of it. Similarly, because we can retrieve the publication times of each item in a feed, we will use this information also

as a feed feature.

On the grounds that none of the approaches described in chapter 2 mention that they use document clustering techniques to improve their algorithms, we chose to include document clustering in our approach. We can assume that the documents of each category of feeds contains somehow similar content, so the feeds can be clustered to meaningful clusters, which we can use to narrow the search space of feeds. With a similar reasoning we chose Latent Dirichlet Allocation and K-Means clustering as our chosen clustering methods. Latent Dirichlet Allocation was chosen because it works well in the field of text document clustering and is easy to see the similarity to human writing process. K-Means was chosen to incorporate a simpler, yet well performing, clustering algorithm.

Although, in this thesis we focus on the RSS recommendation problem, our solution is in no way specific to RSS recommendation. We believe our approach can be used to cluster and recommend any documents that contain the same features we exploited from RSS. For example, to use our approach in recommending scientific papers we could model the authors as a feed source and the papers published by the author as the feed items. With this approach we could get the publication times and content lengths similarly as we did for RSS feeds.

3.2 Text as Data

The data we are working with is text documents, therefore, it is necessary in our case to perform some preprocessing to it in order to improve the quality of the methods applied to it. Many clustering algorithms, including the ones we used in this thesis, are capable of handling numeric data and based on this, we will need to convert textual data to numerical values. The next sections will present an overview to text mining, our preprocessing approaches and our representation of the data.

3.3 Text Mining

Where traditional data mining techniques use numerical data that can be clustered, text mining techniques begins with text data contained in documents. The previous claim might seem to indicate that these fields are separate, however, with nowadays techniques the methods for text mining are more and more similar to the ones used in data mining [47].

To be able to use the traditional data mining algorithms we need a way to

	car	sea	boat	drive
Document 1	1	0	0	1
Document 2	0	2	1	1

Table 3.1: Example of representing text documents with numeric values.

transform the words in a document to a numeric vector. The simplest way to achieve this is to count the number of times a word occurs in a document and then use these counts as the values in the vector. Table 3.1 shows an simple example of representing two documents by counting the occurrences of the words. The example presents two documents where in the first document there are words car and drive, the second document contains sea, boat and drive. Having the data as numeric vectors we can use this information and compute the similarities between these two documents, for example, using vectors space distance measures.

The aforementioned approach allows us to convert the textual documents to a numeric format. Albeit, in this format we have achieved numeric representation of the text documents, we still need to do some processing to get better results as described in next sections.

3.3.1 Text Representation

A well known and widely used vector representation of the documents is term frequency - inverse document frequency (tf-idf) vectors. Tf-idf approach values the terms appearing in the documents so that the most commonly appearing terms throughout all the documents are given less value, and the terms that appear more infrequently are given more weight [47]. The presented valuations can be reasoned with an assumption that common terms do not differentiate the documents well enough, while the terms that appear more rarely are more likely to be specific to the documents in which they appear in.

Equations 3.1 and 3.2 presents how tf-idf value for word w are computed. Equation 3.1 presents inverse document frequency, where N is the total number of documents and $\mathrm{df}(w)$ is the frequency of the word w in documents. Equation 3.2 shows how to compute the value for tf-idf with term frequency, $\mathrm{tf}(w)$, and inverse document frequency, $\mathrm{idf}(w)$.

$$\mathrm{idf}(w) = log(\frac{N}{\mathrm{df}(w)}) \tag{3.1}$$

$$\text{tf-idf}(w) = \mathrm{tf}(w) \cdot \mathrm{idf}(w) \tag{3.2}$$

Tf-idf vectors, which we use to represent the documents, contain the information of how valuable a word is to represent the features of the document. Tf-idf model could be extended to give higher values for different parts of the document [47], for example, if we know that the document's first N words contain more information than the rest, we could give more weight those N words.

Moreover, tf-idf vectors of documents in the basic form still contains words that are not important and can be considered as noise. The next step is to preprocess the documents so that they can perform better in our approach. For a human reader the preprocessed documents might appear of a lesser quality, or even unreadable, but when document clustering algorithms are used with preprocessed documents, they yield better results.

3.3.2 Text Preprocessing

To improve the quality of tf-idf vectors converted from text, we can apply some preprocessing to the text first. We employ natural language processing (NLP) techniques to filter words and convert words to a stem, which is a sort of a root form for the word. The disadvantage of using NLP techniques is, that most of them are language dependent and might require some expert knowledge of the language.

Stop word removal is a technique that removes common words, which might cause some troubles to algorithms and they do not convey any important information. Mostly these stop words in English language contains articles, such as "a", "an" and "the" [47]. Another technique to improve the performance of predictions in text mining is to lemmatize the words. Lemmatization, or stemming, means the conversion of words to their base form [47], to be more specific our technique of choice is called inflectional stemming. For example, *driving*, *driver* and *drove* all have the same root meaning, *to drive*, and we want to use the same root meaning for all the different forms of the word. Regarding the example, we could lemmatize the words to *driv* and replace all the occurrences of the original words with inflectionally lemmatized version of the word. Essentially, this technique reduces the dimensionality of the document vectors also. According to Weiss et al. [47] some approaches in text mining prefer to stem the words all the way to root form. This technique stems the words even further to achieve a deeper generalization, for example, words *teacher* and *taught* could be reduced to *teach* and both of the words would be assumed to have the same meaning.

In addition to the techniques described, we removed short words, of length less than n characters, and converted all words to lower case. Considering the text documents we used, the documents are from internet based RSS

feeds and we need to take care of standardizing the documents.

3.3.3 Document Standardization

For the reason that the sources of the documents can vary in text mining systems the documents should be standardized, this enables us to focus on the information which is useful. The source of a document could be an internet site, a book, an article or any other source of textual information. For example, in a document retrieved from internet the textual content might also include something that is helpful for the web browser to display the content in a more decorated form, as presented in the previous sections, in RSS there exists elements in the content that provide this information. [47]

One example of this kind of widely used format is Extensible Markup Language (XML), or a derived format used in this thesis work called RSS. Both of these formats not only contain the textual information, but also something called tags or elements, which are of no use for the reader but applications can exploit them. In our case we do not need this extra information, we need to identify it and remove it from the documents so that only the contents of the document remains. We used Apache Wink, presented in section 5.2, to retrieve the RSS feeds and remove all the RSS elements from text. To clean the text documents of Hypertext Markup Language (HTML) elements we used JSOUP, described in more detail in section 5.4, which removes all HTML elements from the text.

3.4 Document Clustering

Document clustering is a technique where a large set of documents are grouped together into a meaningful sets of clusters. The grouping can be dependent on the application and it can be, for example, in customer service where reports are grouped into different problems and their solutions. Two main approaches for performing document clustering are hierarchical clustering and flat clustering. The difference between the approaches is that hierarchical clustering contains hierarchy for the clusters and the clustering can be changed by changing the level where the hierarchy is observed. Flat clustering does not contain such hierarchies.

Clustering can be performed in two directions, namely divisive and agglomerative directions. Divisive clustering begins by assuming that the whole set of documents is one cluster and starts to split the cluster to multiple clusters. The process can be continued until each document forms one cluster, or it can be terminated with a stopping criterion [39]. Agglomerative clustering

begins by assuming that each document forms a cluster and combines the most similar clusters together forming bigger and bigger clusters. Agglomerative clustering can also be continued until all documents are assigned to a one big cluster, or it can be stopped at any level with a stopping criterion [39].

Zhao et al. [50] present comparisons on the different approaches of document clustering. According to the results the authors present in their paper, divisive clustering outperforms agglomerative approaches. The authors discuss that the reason for achieving better results with divisive clustering could be due to the possible early mistakes made in agglomerative clustering, which are multiplied as the clustering proceeds.

3.4.1 Clustering Criterion Functions

Clustering criterion functions define which clusters should be considered to be the most similar ones. Depending on the document clustering approach, hierarchical or flat and agglomerative or divisive, there are some criterion functions which perform best.

The paper by Zhao and Karypsis [51] present experiments with some clustering criterion functions for hierarchical clustering. In hierarchical agglomerative clustering there are three strategies to measure the similarity of clusters in order to make a decision which clusters to combine. These strategies most commonly are complete-link, average-link and single-link. There exists also some vector space similarity measures which perform well in document clustering, namely Cosine distance.

For divisive clustering Zhao et al. [50] describe and evaluate some internal, external, hybrid and graph based criterion functions. Internal criterion functions use the documents inside each cluster. External criterion functions use the differences between clusters as a criterion function. Graph based criterion functions model the problem of splitting clusters as a graph and use graph model quality measures. Hybrid models combine all the previously mentioned methods.

3.4.2 K-Means Clustering

The history of K-Means algorithm begun already in 1957, but it was published later in 1982 by Stuart Lloyd [33]. This algorithm is widely used clustering approach because of its simplicity and performance.

K-Means is a simple clustering algorithm for grouping similar objects. The main idea is to choose k cluster centers and then assign each point to the closest cluster center k, this is the assignment step. After all the points are

assigned to a cluster, the centers of each cluster are recomputed, this being the update step, and the process of assigning each point to a closest center is repeated until the update step makes no modifications. Other approach to stop iterating these steps is to define a threshold for the number of changes done in update step. The assignment and update steps can be formulated as follows when the initial set of cluster centroid are $c_1, \ldots, c_k$

- Assignment step,
  - Assign each datapoint x to a cluster of centroid C_i so that the distance to centroid C_i is the minimum distance compared to all other k cluster centroids.
- Update step,
  - Calculate new values for cluster centroids based on the new datapoint to cluster association.
    $c_i = \frac{1}{|C_i|} \sum_{x_i \in C_i} x_i$

One problem with K-Means clustering is how to select the best value for k, the number of clusters. The problem can be solved with preclustering, for example, with canopy clustering to create initial clusters.

3.4.3 Canopy Clustering

McCallum et al. [34] present in their paper *Efficient clustering of high-dimensional data sets with application to reference matching* Canopy clustering. Canopy clustering is a fast single-pass algorithm that can be used, for example, to generate initial clusters as we did. Motivation to use canopy clustering is to reduce the number of comparisons between items in clustering [34]. For example, in K-Means if we had k clusters we would need to begin by computing distances to every cluster centroid from every data point. With canopy clustering we can first run a cheap algorithm to identify the data points that are near to each other and seem to form clusters. When we have created the initial clusters we do not need to compute the distances to each cluster centroid from each data point in K-Means, but rather, we can just compute the distances to the cluster centroid from the data points that are assigned to that cluster by the preclustering algorithm.

Canopy clustering algorithm starts by randomly picking a data point as a center of a canopy and assigns all the data points that are within distance T_2 to that cluster, and at the same time excludes those data points from being part of other canopies. Then all the data points within a distance T_1, from

the previously created canopy, are included in the canopy, but they are also allowed to be part of other canopies.

Modifying the values T_1 and T_2 the number of clusters changes and we can modify the number of the initial clusters with these parameters. According to McCallum et al. [34] canopy clustering reduces the computation time without any loss in accuracy.

3.4.4 Latent Dirichlet Allocation

Blei and Laffert [7] present Latent Dirichlet allocation (LDA) in their paper *Topic models*. LDA is a generative topic model derived from latent semantic indexing and probabilistic latent semantic indexing. The assumption in LDA is that the documents are generated by latent topics, the hidden variables from which the documents are observed from [7]. The main motivation for LDA is to uncover this underlying semantic structure of a text document [7]. These underlying semantic structures are the latent topics of the document and every document is allowed to be a combination of multiple latent topics.

LDA is a generative model and it can be thought to be similar to the writing process of a human. Figure 3.1 shows the plate model for LDA, with the plate model we can observe the similarities to human writing process. The parameters as explained by Blei and Laffert [7] are θ_d in the upper part of the plate model which is the document specific topic distributions, the probabilities of words appearing in topics, α is a positive K-vector used as a parameter for dirichlet distribution for topic proportions. The variable K is the number of topics, D are the documents and N are the words in the dictionary. The topics for words in the topic assignment are in $Z_{d,n}$. Topics are the β_k, multinomials over the whole dictionary, containing the actual words for the generated document. The topics contain all the words in the dictionary so that each topic has some probability for all the words. To generate a document from this model we would begin by choosing one topic distribution for the document, then assigning topics to each word location in the document from this distribution. Finally we would assign words to the locations by drawing them from the distributions of the topics. A more formal description of the process can be found on the paper by Blei and Laffert [7].

Blei and Laffert [7] present the advantages of this variable assignment. It allows each document to contain multiple topics by being a type of mixed-membership model. The key difference between classical mixture models and LDA is that the classical mixture models cannot model this kind of diversity in the topics [7]. For example, if a set of documents is about cars and racing, cars and repairing and third document of racing and repairing.

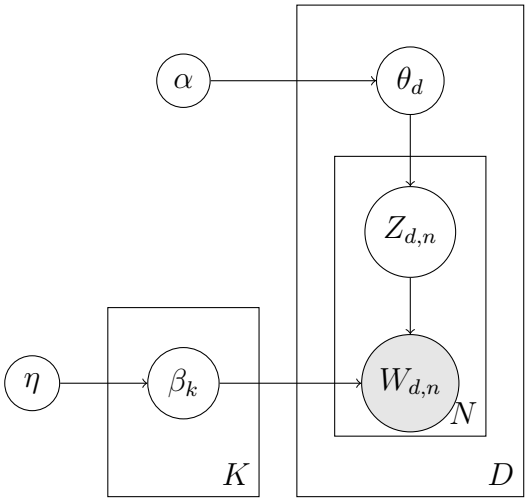


Figure 3.1: A graphical model of Latent Dirichlet allocation as presented by Blei and Laffert [7].

Using classical mixture models the features of cars cannot be captured as well as in a model that allows multiple topics per document.

3.5 Similarity Measures

In order to make reasonable recommendations for users, we need to be able to compute somehow the similarities between the users, between the documents of users and other documents. This section describes the approaches we used in this thesis to measure these similarities.

3.5.1 Vector Space Distance Measures

Because we model the documents as tf-idf vectors we can use many traditional distance measures to measure the similarity of two document vectors. In our approach we have used the following vector space similarity measures:

- Chebyshev,
- Cosine,
- Euclidean,
- Manhattan,
- Minkowski,

- Squared Euclidean,
- Tanimoto.

Chebyshev distance, or L_∞ metric, shown in Equation 3.3 is also known as the chessboard distance [37]. Chebyshev is in the chessboard the distance which the king has to take in order to move to any other location. This is similar to Manhattan distance, with an exception that diagonal moves are allowed.

$$D_{Chebyshev}(A, B) = \max_i (|a_i - b_i|) \tag{3.3}$$

Cosine similarity measures the angle between two vectors [39]. When the angle between the vectors is small the vectors must be similar and the measure is close to one. Similarly, when the vectors are not similar the angle is large and the value approaches zero. Because we want to use cosine similarity as a distance measure we subtract the similarity value from 1 as shown in Equation 3.4.

$$D_{Cosine}(A, B) = 1 - \frac{A \cdot B}{\|A\|\|B\|} \tag{3.4}$$

Euclidean distance measure is a simple distance measure, it measures the connecting line between two points [39]. Equation 3.5 presents the well known approach to euclidean distance in vector space.

$$D_{Euclidean}(A, B) = D_{Euclidean}(B, A) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{3.5}$$

Squared euclidean distance measure is, as the name suggest, a squared distance of euclidean distance as shown in Equation 3.6 [39]. The difference to euclidean distance is that by squaring the distance, we can give more weight to points that are further away from each other.

$$D_{Euclidean^2}(A, B) = D_{Euclidean^2}(B, A) = \sum_{i=1}^{n}(a_i - b_i)^2 \tag{3.6}$$

Manhattan distance, also known as the city-block distance, measures the absolute distance of two points [39]. Equation 3.7 shows the computation of manhattan distance for two vectors. Manhattan distance can be thought to be a stricter version Chebyshev distance, in the same chessboard example only horizontal and vertical moves are allowed.

$$D_{Manhattan}(A, B) = \sum_{i=1}^{n} |a_i - b_i| \tag{3.7}$$

Minkowski distance is a generalization of the euclidean and manhattan distance measures. Minkowski measures the distance in orders of p for two vectors as shown in Equation 3.8. When $p = 1$ the distance equals to manhattan distance and when $p = 2$ the distance equals to euclidean distance. When p has low values we can prevent the distances to become equal with high dimensionality. Minkowski distance gives lower importance to large local distances and makes the distance measure more robust to noise [45]. In our implementation we use $p = 3$.

$$D_{Minkowski}(A, B) = \left(\sum_{i=1}^{n} |a_i - b_i|^p \right)^{\frac{1}{p}} \tag{3.8}$$

Tanimoto distance measure, also known as the Jaccard distance, is similar to cosine distance with a difference that Tanimoto takes into account the relative distance difference of the two vectors [39]. Equation 3.9 presents the formula for computing the Tanimoto distance.

$$D_{Tanimoto}(A, B) = 1 - \frac{A \cdot B}{|A|^2 + |B|^2 - A \cdot B} \tag{3.9}$$

Out of these distance measures cosine distance is used often in measuring document similarity due to its good performance in measuring two vectors similarity. Because cosine distance is widely used in text similarity measures with tf-idf we can use it as a baseline similarity [47].

3.5.2 Shallow Word Ontologies

All the distance measures presented in the previous section compares the vector representations of the documents and the word frequencies in the document. For a human reader, the words have also deeper connections, for example, a banana and an apple are considered to be similar because they both are subconcepts of a fruit. In order to include these connections to similarity computations we used WordNet[1] [36].

WordNet is a broad database of English words, their connections and syntactic categories, from which we are only interested of the connections of words, which WordNet provides as presented in paper by George A. Miller [36]:

[1]WordNet is a registered trademark of Princeton University.

- Synonymy is a symmetric relation between word forms.
- Antonymy is similar to synonymy, but is instead a relation between opposing words and thus is useful for adjectives and adverbs.
- Hyponymy and hypernymy, sub- and super-name respectively, are transitive relations between synsets. These are used for organizing nouns to hierarchical structure. For example, we can use these to get a more generic explanation for a specific term.
- Meronymy and holonymy, part- and whole-name respectively, are complex semantic relations. These are, for example, parts of a machine such as a wheel is a part of a car or ear is a part of a head.
- Troponymy is similar to hyponymy, but for verbs and results in much shallower hierarchical structure.
- Entailment relations between verbs.

WordNet contains at the time of writing over 206000 word sense pairs [46]. There are a lot more features in WordNet, which can be found in WordNet website [46] or in a paper by George A. Miller [36].

We used 8 different word similarity measures to measure the similarity of documents with JIGSAW algorithm for word sense disambiguation [5]:

- Hirst and St-Onge,
- Leacock and Chodorow,
- Lesk,
- Wu and Palmer,
- Resnik,
- Jiang and Conrath,
- Lin,
- Path counting.

Budanitsky [10] gives a good overview of most of the measures listed. Next we will present the similarity measures shortly, more details of the measures can be found in the paper by Budanitsky [10] or in the paper of the original author.

Word similarity named as Hirst and St-Onge is presented in paper by Hirst and St-Onge [25], their approach is based on lexical chains. In Hirst and St-Once two words, or their concepts, are semantically similar if their synsets are connected by a not too long path, also the path should not change direction too often. Leacock and Chodorow uses a method by Leacock and Chodorow [30]. Their method counts the number of edges between word senses in WordNet. The final similarity value is the negative logarithm of the edges scaled by the maximum depth of the WordNet's "is-a" hierarchy.

Lesk similarity is based on the paper by Lesk [31], where the author proposed that the relatedness of two words is proportional to the overlap of their dictionary definitions. Later Banerjee and Pedersen [4] extended Lesk's proposition to WordNet.

Wu and Palmer [49] propose an approach similar to Leacock and Chodorow, earlier than the previously mentioned though. The difference is that Wu and Palmer's approach uses least common subsumer to find the depth of two concepts and scales the result with a sum of the depths of individual concepts, where the depth of a concept is the distance to root node.

Resnik [40] presented a method to measure the similarity of two concepts using the information content. The measure finds the most specific concept that subsumes both concepts, and uses information content to measure the similarity. Information content in this context can be thought as the depth of the concept from root node, the bigger the depth more specific the concept is, therefore, it contains more information than the concepts higher in the tree.

Jiang and Conrath [27] presented an approach to combine the edge based and node based similarities. Their approach starts with edge counting and enhanced with node based approach of information content derived from Resnik's [40] work.

Lin [32] devised a universal distance measure which does not require any knowledge of the items. Lin presents an approach which uses the information content of two concepts to measure the similarity of words. Both Resnik's similarity measure and Wu and Palmer's measure are similar to Lin's measure in certain conditions [10].

Finally the path counting computes the shortest path of two concepts. The value is normalized; if the distance of the path is one the similarity is one and as the path gets longer the similarity approaches zero.

To clarify, all the previously mentioned algorithms compute the similarity of two words. In order for us to use them as a distance metric, we will need to subtract the value of similarity from 1.

3.5.3 Comparing Top-K Lists With Kendall's Tau

Fagin et al. [15] presented in their paper *comparing top-k lists* some approaches to measure the similarity of ordered lists. In the paper the authors use their approaches to measure the similarity of search engine results. The motivation for using the comparison of top-k lists to find similar users is that LDA provides topics which are lists of words. These topic vectors can be ordered by using the probability of the word as a value which is used in ordering. We could also use the centroids of the user's documents as an ordered list of words, and compare the centroids of two users for similarity.

We selected two main approaches from the paper to include in this thesis, which we used with LDA topics. We chose Kendall's Tau and Spearman's footrule distance measures. Spearman's footrule is related to Kendall's Tau in particular $K(\tau_1, \tau_2) \leq F(\tau_1, \tau_2) \leq 2K(\tau_1, \tau_2)$ [13]. In Kendall's Tau the distance measure can be thought as the number of moves needed to order one of the lists to match the other list with bubble sort. Equation 3.10 shows the equation how to calculate Kendall's Tau distance for two lists as presented in the paper by Fagin et al. [15].

The parameter p is the value for penalty which is used when the values of elements in the two lists do not match as explained later, k is the length of the lists, τ_1 and τ_2 are the two lists to be compared and z is the number of common elements in the lists [15]. If we give zero value to penalty p it is an optimistic assignment so that we give non-zero penalty only when we know that the two elements are in opposite order in the two lists [15]. Likewise if we give 0.5 value to the penalty it is a neutral assignment. S is the complement of the two lists $S = D_{\tau_1} \setminus D_{\tau_2}$ and $T = D_{\tau_2} \setminus D_{\tau_1}$. Finally Z is the union of the two lists.

There are four rules which define the penalty for two elements, i and j, compared and these rules can be found in more detail in the paper by Fagin et al. [15]:

- Both i and j appear in both lists. If both items are in same order in both lists penalty equals to zero, if they are in reverse order then the penalty equals to one.

- Both i and j appear in one list, say τ_1 and one of them, say i, appears in other list τ_2. In the former case if i is ahead of j in τ_1 the penalty is zero, otherwise the penalty is one.

- One of the words, say i, appears in say τ_1 but not in τ_2 and j appears in τ_2 but not in τ_1. In this case, the penalty is always one.

- Both i and j appear in one list, but neither appears in the other list. In this case, the penalty is p.

$$K^{(p)}(\tau_1, \tau_2) = (k-z)((2+p)k-pz+1-p)+\sum_{i,j\in Z} \bar{K}_{i,j}^{(0)}(\tau_1, \tau_2)-\sum_{j\in S} \tau_1(j)-\sum_{j\in T} \tau_2(j) \tag{3.10}$$

3.5.4 Comparing Top-K Lists With Spearman's footrule

Other approach we selected to implement from the paper by Fagin et al. [15] was Spearman's footrule. Equation 3.11 shows the formula for calculating the Spearman's footrule distance between two lists τ_1 and τ_2. l is the location parameter for the Spearman's footrule distance which must be greater than k, the length of the lists. As the authors in [15] used value $k + 1$ for l, we used $l = k + 1$ also.

The value of l is used as the position where missing values are inserted to lists. Then basic footrule distance is used to calculate the difference, the first sum in Equation 3.11. The second and third sums are the sums of the elements ranks that are not in the other list.

$$F^{(l)}(\tau_1, \tau_2) = 2(k - z)l + \sum_{i\in Z} |\tau_1(i) - \tau_2(i)| - \sum_{i\in S} \tau_1(i) - \sum_{i\in T} \tau_2(i) \tag{3.11}$$

3.5.5 Hungarian Algorithm

Hungarian algorithm, originally developed by Harold Kuhn [29], is a combinatorial optimization algorithm. The algorithm can be used to find the optimal solution to a pairs assignment problem where the costs of the assignments are known. For example, if there are three workers and three jobs with individual costs per worker, we can compute the assignment of workers to jobs so that the total cost is minimized. It can be thought as a bipartite graph matching.

In this thesis we use it to match the clusters of two users to find the most similar users. When we cluster the documents of all users to same number of clusters, we can compute the distances between the clusters of two users. However, it is not a requirement of the algorithm that we cluster the documents to a specific number of clusters. These distances form the costs for the Hungarian algorithm. When we have the optimal connections

between the clusters that minimize the distances we can sum up all the distances of the clusters and use that as the distance between the users.

Hungarian algorithm computes the distance between users by comparing the clusters, and we can use it as a user distance measure. The algorithm proceeds as follows with an $n \times n$ cost matrix, if the matrix is not square we can add columns or rows with values of maximum value of the matrix. The following listing presents the steps in the Hungarian algorithm.

1. Row by row subtract the smallest cost from all the other costs in that row.

2. Column by column subtract the smallest cost from all the other costs in that column.

3. Cover all the zeros in the matrix with a minimum number of lines.

4. Test for optimal solution:

   (a) If the minimum number of lines from step 3 is n we have found our optimal solution. The optimal solution are the zero values so that each row and each column has only one zero selected.

   (b) if the minimum number of lines from step 3 is less than n, continue to step 5.

5. Determine what is the smallest cost not covered by any line and subtract it from all rows that are not covered. Add the smallest cost to all columns not covered and return to step 3.

3.6 Recommender Systems

Recommender systems can be thought to be derived from the human behavior in decision making. Usually when a person is making a decision from a set of similar products, he or she asks from his or her friends for recommendation on how to make a decision. Another solution could be to find a similar person based on preferences and trust this person's recommendation. In a similar manner recommender system is given a set of items and a user model, be it either the social connections of the user or the preferences. From this information the recommender system computes the most appropriate items to help make the decision. These kinds of problems, where a decision must be made, are common nowadays with the expansion of Web based stores, where the selection of items can be broad [41].

We can express the decision making problem as an information overload problem. Depending on the domain there can be hundreds of relevant items to choose from. Finding the interesting subset of these items can be a tedious task and making the educated decision of the item to choose can be also difficult. Recommender systems aim to help make the decision by exploiting the knowledge of the user's preferences.

In addition to traditional recommendations needed for items to purchase, there has been many different fields to apply recommendation systems in the past. Perhaps the most widely known problem is the Netflix Prize [6], where a movie recommender system is developed. The Netflix Prize competition has given a great deal of publicity for collaborative filtering techniques, where the recommendations are based on the valuations of other users of the movies they have seen.

Recommender systems are software systems built with techniques that are able to recommend items to users that the user might be interested of. Items in the context of recommender systems can vary from news items to a product in a store. There are personalized and non-personalized recommendations from which the latter is not usually interesting in the field of recommender systems [41]. Non-personalized recommendations are typically seen in magazines and in other static content where the recommendations are made for a broad audience, in these cases the recommended items typically are top lists of items [41].

In the book *Recommender Systems Handbook* by Ricci et al. [41] the introduction chapter presents an overview of personalized recommendations. Personalized recommendations are, in the simplest case, presented in an ordered list where the most interesting item is the highest ranked item. To be able to recommend items to a specific user, some sort of user profile or model must be learned. User profile usually contains user's preferences which can be collected explicitly by collecting user's valuations of items. Other approach to collect user preferences is to infer the ratings from user actions, such as a user buying an item or viewing an item description, which can both be interpreted as an interest towards the item [41].

Recommender systems can be divided to two main approaches in recommendation: user based and content based recommendations. The main difference is the data which is used to make a recommendation. In user based recommendations, or collaborative filtering, there is no need for knowledge of the item being recommended, only the user profiles built from item valuations are compared [41]. The basic approach is to first find the most similar user to the user for whom the recommendations are presented to, the target user. Then these two similar users are compared so that the items for which the target user does not have any information, are recommended in an order

	item 1	item 2	item 3	item 4
user1	5	5	1	-
user2	-	1	3	4
user3	-	1	3	-

Table 3.2: Example of user based recommendation.

defined by the likings of the similar user.

Table 3.2 presents a simple case of three users, from which we would like to recommend an item to user 3. We start by finding the most similar user, and discover that user 2 is the most similar to user 3 by examining their likes on items. Then we search for items that user 2 has rated high and user 3 has no rating, we can observe that item 4 is such an item.

Content based recommendations are decided based on the similarity of items. Content based approach is domain specific and requires knowledge of the items and the properties of them. For example, in a recommendations for books the author, title, year and publisher could be interesting properties. With this information we could explore user's favourite books and might discover that publisher X is dominant in the favourites, and we could recommend books from publisher X to the user. The problem with content based recommendations is that they are very domain specific and a book recommender cannot be used for an e-commerce site selling gardening products. This is the reason why most of the available libraries and frameworks for recommender systems are focused on user based recommendations [39].

Herlocker et al. [24] present in their paper a set of domain-independent tasks or goals usually performed by recommendation system, or expected to perform. The following listing presents the tasks as presented in the paper:

- Annotation in Context: Highlight items in a list to emphasize them.
- Find Good Items: Create a ranked list of recommended items with predicted ranking.
- Find All Good Items: Recommend all somehow relevant items, this can be used, for example in medical recommendations to make sure no option is overlooked.
- Recommend Sequence: Recommend a sequence of similar items, for example, a sequence of research papers, a tv show and so on.
- Just Browsing: Help the user in the browsing experience to create a pleasant browsing experience.

- Find Credible Recommender: Users might not initially trust the recommender and might want to play with it first to create a trust.
- Improve Profile: Users rate items because they believe that improves the quality of the recommendations.
- Express Self: Users might not be interested in the recommendation but are more interested in expressing themselves using their ratings.
- Help Others: This is similar to previous task, users might want to help others to find good recommendations and thus rate items.
- Influence Others: To increase the publicity of an item, a movie for example, movie studios might increase the rating of movie to influence viewers to go see the movie in theaters.

These tasks define the end user experience of a recommender system and should be taken into consideration when designing a recommender system.

Chapter 4

The Proposed Approach

In this section we will present the proposed approach for our problem. We begin by describing the problems we are solving and how we solved them, we continue with the document clustering techniques and explain our approach in recommender system and the similarity measures we developed.

4.1 How We Approach the Problem

To help the reader to understand our approach, we begin by introducing the problems we are solving. First we must model the user, find the feeds to recommend and finally order the feeds in an order that presents the most likely interesting feeds first.

We have a simple approach to model the user's interests; we model the user based on the content discovered from the feeds the user subscribes. This allows us to modify the user model easily with weighting approaches, such as favourite weighting and content dependent weighting. Once we have an accurate model of the user's interests, we have two approaches to find the set of feeds to recommend; we can use purely the user model and find feeds that are similar to the user model, or we can compare the user models and recommend the feeds that the most similar user has subscribed.

Once we have discovered the set of similar feeds we compute the distance of the feed to the user model depending on the approach. If we use user based recommendation we compute the distance to the user model, or if we use content based recommendation we use the distance to the individual feeds of the user. Once we have computed the distances of all the feeds we discovered in previous step, we can order them based on the distance to user and recommend the top-5 most similar feeds.

We could have chosen to model the user differently, for example, we could

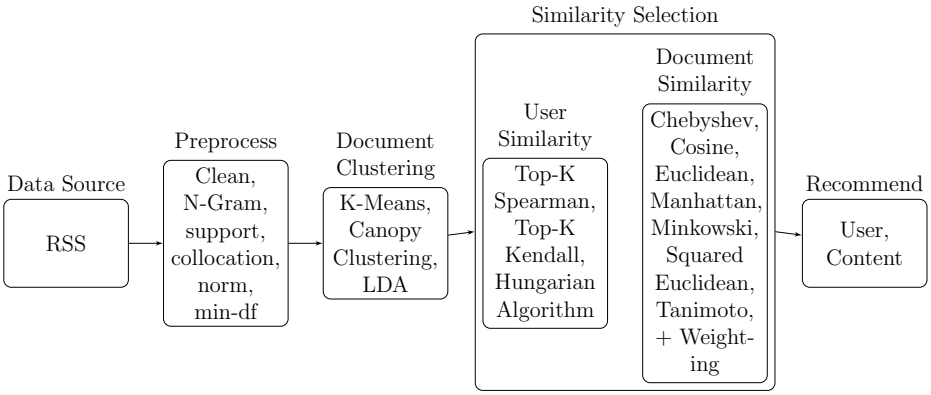


Figure 4.1: Module diagram of our application.

have used the explicit information of the user's preferences and find similar users. However, this approach requires explicit input from the user and we did not want to require this from the user, thus we can collect implicit information of the user's interests by using the content. We included favouring the feeds in order to weight the similarities of feeds, but we did not want to make it the main approach to recommend. Another approach would have been to use social networks and use the connections of users, this way we can get the feeds from the users that are connected. This approach assumes that the connected users share their interests in order to make good recommendations, we did not want to build our approach based on this assumption.

Figure 4.1 presents the modules in our applications, it can help the reader to understand our approach better if we briefly describe it here. Our approach can be separated to 5 modules, we have so far described data source, preprocessing, document clustering and a subset of the similarity module. We present in the following sections our approaches in the remaining parts of the similarity module and the recommendation module.

4.1.1 Baseline method

For comparison purposes we developed a baseline method which counts the number of shared feeds between users. The shared feeds are used to find the most similar user and the feeds of the most similar user are recommended to the user. This is far more simpler than any other method we used in this thesis, and can be used to compare against the more developed approaches.

Baseline method is a hybrid of content and user based approaches. We first use the number of shared feeds to find the most similar user and then compute the distances of the similar user's feeds to the target user's feeds,

finally we use these distances to select the top-5 most similar feeds to recommend.

4.1.2 Collaborative Filtering

Collaborative filtering is a recommendation approach widely used and a very flexible method, it does not need any information about the features of the items that are being recommended. Collaborative filtering uses the user's valuations of the items to compute the distances between users. One disadvantage of this approach is that in order to work properly, it needs the information of how much the user is interested of each item.

There are approaches that can compute the user interest in item explicitly and implicitly, by asking the user to rate or by following the user behavior respectively. In our approach we used only explicit information of the valuation of the feed, which we can use in collaborative filtering.

We treat collaborative filtering as a baseline method, it is similar to our previously presented baseline approach, but it uses the user's favourites as a similarity measure instead of content similarity. Collaborative filtering is generally considered to be a good approach to recommend items, for example, in Netflix competition collaborative filtering performed well [28]. In the next sections we present the methods we consider to be more sophisticated, and these methods contain the main research and development done in this thesis.

4.2 Semantically Generalized Documents

The previous chapter presented some WordNet based similarity measures for words. In addition to measuring the semantic similarities between words, we used WordNet relations to generate semantically generalized documents. Semantically generalized documents can be generated by converting top N words of each document to hypernyms of the words, which are the general terms for the word. For example, a specific term can be an apple and the first general term for this word is fruit. This approach attempts to make the distance measurements between documents more generalized, by using more general terms for words instead of specific words.

For example, consider a document about specific dog species and another document about specific cat species. With these detailed terms, the distance measure would find the documents quite dissimilar. By converting the detailed words to a more general terms the distance between these documents is smaller, as it should be. The idea behind this is that these documents both are about pets and should perhaps be considered similar.

The reason for using only the top N words appearing in the documents is chosen because computing the hypernyms for words from WordNet is computationally heavy, also the most dominant words in a document are enough to distinguish the general topic of the document.

4.3 Recommendations

Recommendation approaches used in this thesis can be divided to two main approaches, one is to measure the similarity by using the content of the feeds. The other approach is to compare the features of users to find similar users and use that information to recommend content.

Next we describe these two approaches in more detail and how they are used with different document clustering algorithms. We begin with content based similarity measures.

4.3.1 Content Based Similarities

Content based similarities use the textual content of the feeds as the main source of information. In this thesis there are two implementations of content based similarities, one for K-Means and one for LDA.

These data sources are quite similar as they both produce vectors representing some set of similar documents that are grouped together. In K-Means these vectors represent the centroids of the clusters, similarly in LDA these vectors are the topics which are assigned to each document.

4.3.1.1 K-Means

In K-Means approach the centroids of each cluster is used to find similar documents. Our approach is to find all the clusters that the user has documents in and use all the user documents in each cluster to compute a user specific centroid for that cluster. We compute the distances from this user specific centroid to other documents in that cluster, that are unknown to the user. Algorithm 1 presents the process of finding the closest documents to the user's documents.

Now we have the distances of all the feeds that are unknown to the user. With this information we can order the list of unknown feeds by distance and recommend the feeds that are the closest. We could also compute the distances of all documents against all the user's documents to find the globally most interesting documents. However, this can be computationally too

expensive, if and when the number of documents is large. Limiting the number of document comparisons to already known similar subsets of documents can decrease the computational cost.

Algorithm 1 K-Means content based recommendation

```
for all clusters C do
    for all user documents D in cluster C do
     compute centroid c_m for C from documents D
    end for
end for
for all centroid c_m do
   find closest non-user document d_new in cluster C from centroid c_m
end for
order the list of d_new by distance
```

4.3.1.2 Latent Dirichlet Allocation

In content based recommendation with LDA clustered documents, we begin by searching all the topics that the user has feeds from. Vectors representing the topics, we can discover the most similar topic for each user topic, and add those to the list of topics that the user might be interested. Now that we have a set of interesting topics for the user we can search for all the feeds in these topics that the user does not have and compute the distances to the user's topics. We select the recommended topics by the distance from the topics that the user is already interested of.

Having discovered the similar topics we can search for all the documents in these topics and compute the distances between the user's documents. The distance is computed using the tf-idf vector of the documents and with different distance measures. With the same reasoning as in K-Means content based recommendation, presented in algorithm 1, we limit the number of document comparisons by finding first an interesting subset of the documents. Algorithm 2 presents our approach in computing content based recommendations for LDA.

4.3.2 User Model Based Similarities

The main motivation for using user based similarities is that if the two user models are similar, it is probable that the actual users also like similar topics. To measure the similarity of users we used K-means and LDA based

Algorithm 2 LDA content based recommendation

for all user topics T **do**
  find the most similar topic to T and add to the interesting topics $T_{interest}$
end for
choose the top-3 most similar topics from $T_{interest}$ to T_{close}
for all topics in T_{close} **do**
  find all documents D_{new} in T_{close} that user does not have and compute distance to user topics
end for
order the list of interesting document D_{new} by the distance to user topics

approaches. In K-means the user models are computed from the cluster information and in LDA the user models are computed from the topics. One traditional approach in user based recommendations is to use collaborative filtering to find similar users. This approach is different from the previously mentioned, as it does not need any knowledge of the items to recommend and is more flexible.

4.3.2.1 K-Means

K-Means generates clusters of the documents, which we can analyze and form a user model based on the clusters. To model a user in K-Means approach, we use all the documents that the user has and compute centroid from them. Because the documents are modeled in vector space, we can use vector centroid of all the user documents as a user model. Equation 4.1 presents the approach to compute the user model C_{user} from the user's documents D_{user}.

$$C_{user} = \frac{1}{|D_{user}|} \sum_{i \in D_{user}} D_i \tag{4.1}$$

Once we have computed the user models for all users we can compute the most similar user by comparing the user models with some traditional vector distances described in subsection 3.5.1. We exploit the clusters of the most similar user and retrieve all the documents in these clusters that are unknown to the user we are making recommendations to. To order the documents that we have found, we use the target user's feeds to compute the distance to each new feed, and use the smallest distance as the distance of the new feed to user. Algorithm 3 presents the described approach.

Algorithm 3 K-Means user model based recommendation. U_{target} is the user we are making recommendations to.

for all user u in users U **do**
  compute user feed centroid C_u
end for

for all user feed centroids C_u **do**
  compute distance to U_{target}
end for

for all feed clusters c_{feed} in closest user clusters **do**
  get all unknown feeds $feed_{unknown}$ to user U_{target} in
  closest users clusters c_{feed}
end for

for all feed $feed_{unknown}$ **do**
  compute smallest distance to all
  user feeds $feed_{user}$
end for
order recommended feeds by distances

4.3.2.2 Latent Dirichlet Allocation

In K-Means user model based approach we computed the user model from the user's documents, similarly in LDA we compute the user model from the topics of user's documents. Equation 4.2 presents the approach to compute a user centroid based on the topics of the user's documents. Approach that we developed uses only the most probable topic for the feed. This simplification allows us to convert the soft document to topic association generated by LDA to hard assignments.

$$C_{user} = \frac{1}{|T_{user}|} \sum_{i \in T_{user}} T_i \tag{4.2}$$

We can compute the most similar users by computing the distances between users with vector space distance measures presented in subsection 3.5.1. Next we used the topics that the most similar user is interested of in order to find feeds to recommend. We collect all the feeds that are in the topics of the most similar user and order them by computing the distance of the feed to all of the topics that the target user is interested of. Algorithm 4 presents our approach to compute the user based recommendations for LDA, the al-

gorithm is similar to the one used for K-Means user based recommendations.

Algorithm 4 LDA user model based recommendation. U_{target} is the user we are making recommendations to.

for all user u in users U **do**
  compute user topic centroid T_u
end for

for all user topic centroids T_u **do**
  compute distance to U_{target}
end for

for all feed topics T_{feed} in closest user clusters **do**
  get all unknown feeds $feed_{unknown}$ to user U_{target} in
  closest users topics T_{feed}
end for

for all feed $feed_{unknown}$ **do**
  compute smallest distance to all
  user topics T_{user}
end for
order recommended feeds by distances

4.4 Feed Weighting

To improve the quality of the recommendations, we implemented four different feed weighting approaches that model the quality of the feed, user valuation of the feed and the semantic similarity of the content in the feeds. Our approaches were chosen based on the performance in other papers, and our own intuition what could help to improve the quality of the recommendations by improving document similarity measures. In the next sections we describe our methods in more detail.

For three of the weighting approaches, all except favourite weighting, we gave a total of 25% effect to distance measures from weighting. If only one weight was used that weight controls the whole 25%, and if all three weights were in use each of them controls 8.3% of the effect. Equation 4.3 presents the formula for computing the total distance for feed f, where $dist(f)$ is the distance value for the feed using the selected vector space similarity measure.

Parameter *weights* is the sum of all enabled weights, for example, sum of inverse update frequency and content length.

$$dist_{total}(f) = 0.75 \cdot dist(f) + 0.25 \cdot weights \tag{4.3}$$

4.4.1 Favourite Feeds

Weighting the user's favourite feeds by allowing the user to like their favourite feeds are quite common approaches in many recommendation systems. This approach is simple and it also allows us to implement collaborative filtering.

We implemented favouring in such a way that allows user to only like or set the liking to default. One more option would have been to dislike feeds but for this thesis we did not implement disliking. We used three numeric values for likes; value 1 for like is the default and values 2 and 3 are the two like settings.

We used the like values when computing the user models. We also included favouring to content based approaches, which are based on K-Means clusters or LDA topics, by multiplying the document vector with a value set for like. This allows the user model to move slightly to the direction of more favoured feeds.

$$C_{user} = \frac{1}{|D_{user}|} \sum_{i \in D_{user}} D_i \cdot like \tag{4.4}$$

4.4.2 Inverse Update Frequency

Inverse Update Frequency (IUF) was introduced by Ji and Zhou [26] in their paper *A Study on Recommendation Features for an RSS Reader*. Based on the authors experiments and the results presented, IUF was the best performing feed weighting approach when combined with favourite fraction. Favourite fraction is the ratio of favourite items in a feed with respect to the items read in the feed, as we do not follow the favourites in item level, we used only IUF.

Equation 4.5 presents the computation of IUF, where $n(i)$ is the average number of new items per day for feed i and n_{max} is the maximum number of updates per day from all feeds. IUF models the quality of the feed so that it is assumed that if a feed is updated too frequently the quality of the feed is low and with less frequent updates the quality of the feed increases.

$$IUF(i) = \frac{n(i)}{n_{max}} \tag{4.5}$$

IUF is not user specific, instead it is a feed specific and we use it to weight the distance computation for the ordering of the recommended feeds. In Equation 4.5 $IUF(i) = 0$ when the feed is the most infrequently updated, and $IUF(i) = 1$ when the feed is updated frequently. With these values we can increase or decrease the distance value computed.

4.4.3 Content Length

With a similar intuition as with IUF, we used the length of the feed content to measure the quality of the feed. The main difference between the IUF and content length is that we prefer longer contents so that the feed contains items that are more descriptive. If the content length is short the item might not contain enough information to make a decision if it is worth reading or not.

Equation 4.6 presents the calculation of content length for feed i, where $l(i)$ is the average length of content for items in feed and l_{max} is the maximum length of content from all feeds. Similarly as IUF, content length is used to model the quality of the feed and it is used when ordering the feeds for recommendation. When $CL(i) = 0$ the content of the feed contains many characters and describes well the feed contents. When $CL(i) = 1$ the content of the feed is short and we can assume the quality of the feed is not so good.

$$CL(i) = 1 - \frac{l(i)}{l_{max}} \tag{4.6}$$

4.4.4 WordNet Title Similarity

Often the title of the feed is a good description of the feed's content and of the topic that the feed covers. With this intuition we implemented a feed similarity measure which can be used as a feed weighting approach. Our approach uses WordNet to compute the similarity of the words found in the feed title.

To use this similarity as a feed weight, we compute the distance of a recommended feed's title to all the words in user's feed titles. By using all the words in user's feed titles, we can get a more general model for user's interests and compare that against the new feed. The similarity measures we used to compute the similarity are presented in subsection 3.5.2. When we compute the similarities of the titles, we need to convert this to a distance by subtracting the similarity from 1. We can then use this value similarly as content length and IUF.

Chapter 5

Implementation Environment

This chapter explains in brief detail the software packages we used in our implementation. We chose to use Java as the programming language to implement the application, because Java environment provides us good tools for our task. These tools include projects from Apache such as Mahout, Wink and Lucene to mention a few. All of the tools used are open source and are under active development, which means that the algorithms are more likely to be up to date with the most recent and well implemented algorithms. In the next sections we will describe these tools in more detail and finish this chapter with a module level explanation of our tool.

5.1 Apache Mahout and Hadoop

Mahout is a machine learning library from Apache which provides algorithms to recommendation systems, clustering and classification [39]. To make Mahout scalable the developers have built it using Apache Hadoop in the core of some of its functions. Hadoop is a framework for distributed computing and more information of it can be found on the project's website [1].

Mahout itself is only a software library written in Java, which means that it is not usable as a standalone tool. It does not provide any graphical user interface and it requires some development time to use it in any project. In this thesis we used clustering algorithms and vector space distance measures from Mahout.

There would have been many alternatives to Mahout, even if we would still use Java. Some alternative libraries include Weka [22] and Mallet [35]. Mainly due to the good integration to Hadoop we chose to use Mahout as our library of choice. Mallet would have been another good choice as it is focused on algorithms for textual content.

5.2 Apache Wink Client

Apache Wink is a framework for using web services from Java code [3]. Wink provides an easy access to web services and as a developer, one does not need to know all the low-level aspects of communication with a service. This framework allowed us to easily consume RSS feeds in different forms so that we could only focus on the information contained in the feeds.

In addition to Apache Wink there are numerous web service libraries for Java and it does not make a big difference which one we chose as we barely required any features from it.

5.3 Apache Lucene

Apache Lucene is a long running project providing search framework [2]. Apache Mahout also used to be part of Apache Lucene, but as the machine learning capabilities got bigger in the project, Mahout was separated to its own project [39].

Apache Lucene provides indexing and search technologies and text analysis tools, just to mention a few. We only scratched the surface of Lucene in this thesis, as we only used the text analysis it provides. We used Apache Lucene to perform the text preprocessing by implementing our own filter for Lucene. This filter can be used to generate vector representations of the documents with Apache Mahout. Due to our choice of using Mahout and its tight integration with Lucene, we did not see it necessary to look for alternatives. If we would have built our application using, for example, Mallet it would have been reasonable to look for alternatives.

5.4 Jsoup HTML Parser

Jsoup is a Java HTML parser which can do many things with related to HTML code in textual format [23]. We also used only a fraction of the tools capabilities, as we only use it to parse the text from the RSS feed items.

Usually there are some HTML code in RSS feed items so that, for example, browser knows how to style the feed contents. For us this information is not useful and instead it is more harmful, because HTML is universal language and the same syntax is used in all feeds. Similarly as for Apache Wink, there are numerous HTML parsers for Java and any of those would have been just as suitable for our needs.

5.5 WordNet Jigsaw

Jigsaw is a knowledge-based word sense disambiguation (WSD) system presented by Basile et al. [5], which uses the WordNet senses to disambiguate. The main assumption in Jigsaw is that part-of-speech specific algorithms are better than using a general approach for all word classes. For example, Jigsaw uses different approaches to disambiguate senses for adjectives and verbs [5].

We chose WordNet Jigsaw based on the algorithms it includes and that they are all well documented and implemented. There are many WordNet libraries available for Java and for our application the WordNet was not the main focus, therefore, we did not see it necessary to experiment with different libraries.

5.6 Our Application

Figure 5.1 shows the high-level implementation of our application. There are 5 main modules in the application which can be divided to data source, preprocessing, document clustering, similarity and recommendation modules.

In more detail Figure 5.2 describes the connections between the different libraries in our application. Starting from the top left corner of Figure 5.2 Wink consumes RSS feeds and outputs the information contained, all RSS elements removed, to JSOUP which removes all HTML elements from the text. The processed text files are stored to file system for persistence. Mahout uses this data in file system and preprocess the files using Lucene and writes the analyzed results to file system. Mahout uses the contents in the file system to perform clustering using Hadoop, and finally outputs the results to file system and to application for computing recommendations. Finally, the results are outputted to graphical user interface.

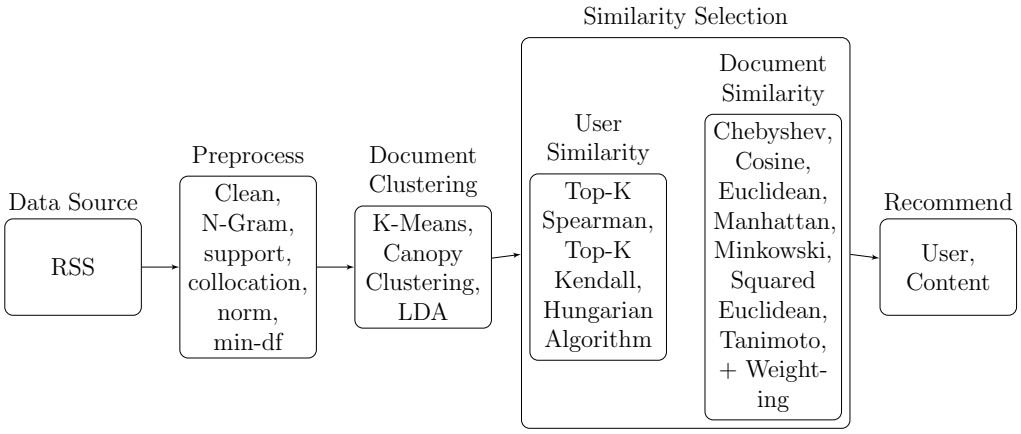


Figure 5.1: Module diagram of our application.

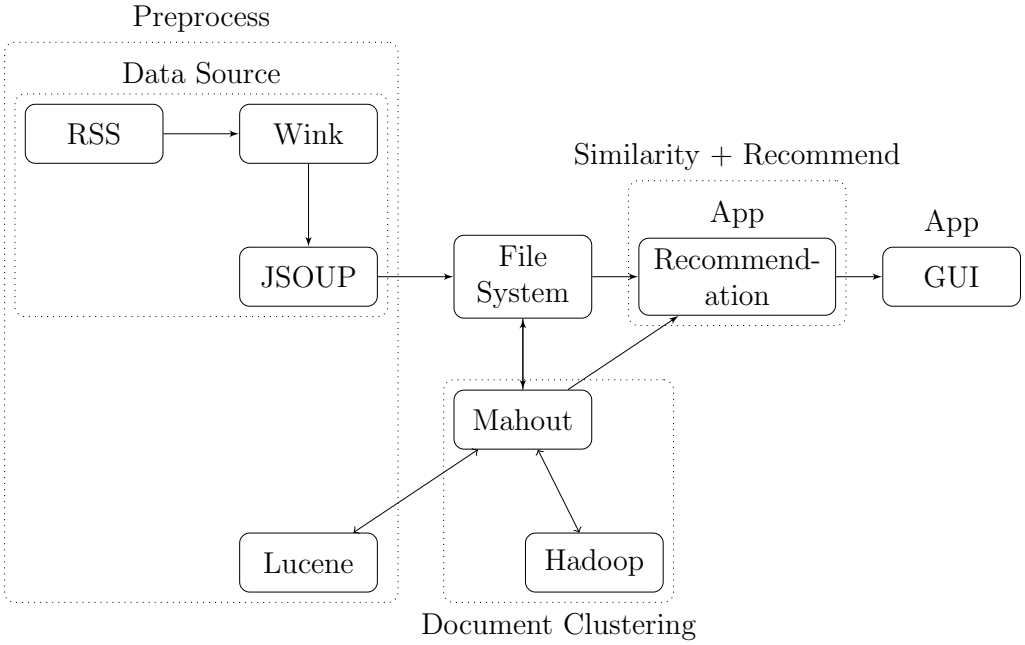


Figure 5.2: Block diagram of our application. Dotted areas denote roughly the modules shown in Figure 5.1.

Chapter 6

Experiments and Results

This chapter presents the experiments we conducted on our approach and the results. We begin by describing our dataset, how we gathered our feeds and continue by presenting our experiments and results for text preprocessing. We continue with the clustering experiments and results, then we present our recommendation experiments with three different datasets and the results from those. Finally, we experiment with distance measures of the documents and the user models.

6.1 Dataset

There are some recommendations for a dataset to be used in recommendation system evaluation, for example, in collaborative filtering it is recommended that there are multiple items, multiple ratings per item and that there are more ratings than items [44]. We took these recommendations into account when we constructed the datasets for our experiments.

Our source of feeds that we used to experiment with contained about 1200 feeds available on the internet. These feeds covered many general categories such as art, business, entertainment, health, science and hobbies. The full listing of feed categories can be found in Appendix A. The feeds were mainly collected from two feed collections freely available online, Feedage.com [16] and Feeds2Read.net [18]. We also collected feeds from popular news sites such as BBC and NY Times.

6.1.1 Generated Users

We devised a few sets of different users for our experiments. These datasets can be divided to two categories; large datasets of a hundred users and small

datasets of a few tens of users. Large datasets were used to compare the different clustering approaches and automated experimenting of recommendations. Smaller datasets were used to conduct experiments with recommendations and a human interpretation of results was used on those. For recommendations experiments the smaller dataset was used because the results had to be evaluated by hand, and using hundred or more users and many different algorithms would be too time consuming. Some evaluation metrics can be automated, such as comparing the results with categories of the feeds, but this is not as accurate as evaluating feed interests manually, as shown later in our results.

6.2 Preprocessing Text

We experimented with preprocessing parameters to find the optimal values that produce the best clustering results. We experimented with the following parameters for text preprocessing:

- Term length filtering to filter out short terms,
- Maximum allowed n-gram,
- Minimum support for term,
- Minimum log-likelihood ratio for collocations,
- Norm to use,
- Minimum document frequency.

We measured the effect of these six parameters by evaluating the cluster quality of the generated clusters. These quality measures are explained in more detail in the beginning of section 6.3. We used K-Means and LDA clustering results in our experiments and optimized parameters for these. According to our experiments the best parameters for text preprocessing were found to be the ones presented in Table 6.1.

The detailed results of our experiments are presented in Appendix B, where we can observe the effects of the parameters. If we do not filter out short words the average inter cluster distance is small and the number of clusters is small, which suggest that keeping all the words produces noise to clustering and these short words are shared among the documents. With our chosen value 5 for minimum word length we can increase the number of

Parameter	value
Term length	5
Maximum N-Gram	4
Minimum support for term	1
Minimum Log-Likelihood ration for collocations	10
Norm	1
Minimum document frequency	2

Table 6.1: Results for optimized text preprocessing parameters.

clusters and the average inter cluster distance is increased, also CDbw value is higher with filtering out short words.

For N-Gram results we can see that increasing the allowed N-Grams increases the number of clusters and CDbw value, however, allowing too many clusters to form is not good for finding feed groups. From intra cluster density we can see that there is a maximum density at 4-Grams, and also inter cluster densities are at this value at minimum, these results indicate that this would be a good choice as the clusters are dense and do not overlap.

Increasing the minimum support for term decreases the CDbw values and intra cluster densities, which gives a reason to keep this value low. Minimum log-likelihood ratio for collocations has no effect with unigrams, therefore, we experimented with bigrams for this to have an effect. We chose the value for minimum log-likelihood ratio for collocations based on the maximum intra cluster density and maximum CDbw value, which both occur at value 10. For the norm to use we used the same criteria as for the minimum log-likelihood ratio for collocations and found the best value at 1. For minimum document frequency the results got worse as the value was increased and thus we selected the value of 1 for minimum document frequency.

6.3 Document Clustering

The results of clustering algorithms affect the quality of recommendations as our recommendation algorithms rely on the document cluster associations. We measured the quality of both K-means and LDA clustering with the same metric called CDbw. Next we will present the CDbw measure presented by Halkidi and Vazirgiannis [21].

6.3.1 CDbw

Halkidi and Vazirgiannis [21] present a measure called compose density between and withing clusters (CDbw) in their paper. Their measure uses inter and intra cluster densities to compute value for this measure. Equation 6.1 shows the formula for computing CDbw value for cluster index c where separation is computed with minimum distances between clusters and the inter cluster densities as shown in Equation 6.2, where d_i and d_j are the items in clusters [21]. Inter cluster density and intra cluster densities are explained in the following sections 6.3.3 and 6.3.2 respectively.

$$CDbw(c) = Intra\_cluster\_density(c) \cdot Separation(c) \tag{6.1}$$

$$Separation(c) = \frac{\sum_{i=1}^{c} \sum_{\substack{j=1 \\ j \neq i}}^{c} min\_dist(d_i, d_j)}{1 - Inter\_Dens(c)}, c > 1 \tag{6.2}$$

6.3.2 Intra Cluster Density

Intra cluster density measures the number of items in a cluster that are located within the average standard deviation, average being computed from all clusters, from the center of the cluster. This measure should be maximized as it informs us how dense the cluster is, if the value is low we can suspect that the cluster is sparse and this might lead to overlapping clusters or might indicate that the number of clusters should be higher.

Equation 6.3 shows the intra cluster density as described by Halkidi and Vazirgiannis, where v_{ij} is the j datapoint in cluster i [21]. Equation 6.4 presents the density function used in Equation 6.3, which counts the number of items in the cluster within one standard deviation distance from the center with Equation 6.5.

$$Intra\_Cluster\_Density(c) = \frac{1}{c} \sum_{i=1}^{c} \frac{1}{r} \sum_{j=1}^{r} \frac{density(v_{ij})}{stdev}, c > 1, c \neq 0 \tag{6.3}$$

$$density(v_{ij}) = \sum_{i=1}^{n_i} f(x_i, v_{ij}) \tag{6.4}$$

$$f(x, v_{ij}) = \begin{cases} 0, & \text{if } d(x, v_{ij}) > stdev \\ 1, & \text{otherwise.} \end{cases} \tag{6.5}$$

6.3.3 Inter Cluster Density

The inter cluster density tells us how much the clusters overlap. This measure is computed by comparing the clusters pairwise and computing the center point for the two clusters and the average standard deviation from these clusters. The density is computed from the items that are located within one standard deviation from the center point of the clusters. Intuitively, this measure should be minimized to get as small overlap between the clusters as possible. Equation 6.6 shows the equation for computing inter cluster density as presented by Halkidi and Vazirgiannis [21].

$$Inter\_Dens(c) = \sum_{i=1}^{c} \sum_{\substack{j=1 \\ j \neq i}}^{c} \left(\frac{dist(c_i, c_j)}{stdev_i + stdev_j} \cdot dens_{inter}(u_{ij}) \right), c > 1, c \neq n \tag{6.6}$$

Equation 6.7 measures the number of points in the radius of average standard deviation from the center point of the two clusters. Equation 6.8 presents the function which is used to count the items which are close to each other, similarly as in Equation 6.5. This value is divided by the total number of items in both clusters in order to get the proportion of the shared items.

$$dens_{inter}(u_{ij}) = \frac{\sum_{l=1}^{n_i+n_j} f(x_l, u_{ij})}{n_i + n_j} \tag{6.7}$$

$$f(x, u_{ij}) = \begin{cases} 0, & \text{if } d(x, u_{ij}) > (stdev_i + stdev_j)/2 \\ 1, & \text{otherwise.} \end{cases} \tag{6.8}$$

6.3.4 Inter Cluster Distances

The three inter cluster distances that we measured were minimum, maximum and average distances. The most important of these three is the average distance between clusters, which measures similar value as the inter cluster density, it tells us how far apart the clusters are from each other. The minimum inter cluster distance is the smallest distance between any two items of two clusters and vice versa the maximum distance is the largest distance. Average distances are computed by measuring the distances between all items in two clusters and taking the average of this value.

6.3.5 Approach for Latent Dirichlet Allocation

Because LDA creates a soft clustering of the documents, the aforementioned metrics cannot be directly applied there. Our approach to measure LDA results with CDbw and cluster densities takes into account also the probability of each topic association for document.

To compute the intra cluster density for LDA clustering we use the document distance between documents A and B and multiply the distance with the probability of both documents being assigned to topic t. Equation 6.9 describes the approach to compute intra cluster density for LDA, where P_{A_t} and P_{B_t} are the probabilities of topic t for documents A and B.

$$LDA\_intra\_cluster\_density(t) = \sum_{A}^{D} \sum_{\substack{B \\ B \neq A}}^{D} dist(A, B) \cdot P_{A_t} \cdot P_{B_t} \tag{6.9}$$

Similarly for inter cluster distances we also take the probabilities of document being assigned to a topic into account. Equation 6.10 shows the approach to computing inter cluster distances.

$$LDA\_inter\_cluster\_density(s, t) = \sum_{A}^{D} \sum_{\substack{B \\ B \neq A}}^{D} dist(A, B) \cdot P_{A_s} \cdot P_{B_t} \tag{6.10}$$

The separation, shown in Equation 6.11, is also modified for LDA so that the minimum distance is multiplied by the probability of the document being part of that topic cluster.

$$Separation(t) = \frac{\sum_{i=1}^{t} \sum_{\substack{j=1 \\ j \neq i}}^{t} min\_dist(d_i, d_j) \cdot P_{d_{it}} \cdot P_{d_{jt}}}{1 - LDA\_inter\_cluster\_density(t)}, t > 1 \tag{6.11}$$

6.3.6 Results For Clustering Algorithms

Our approach to experiment with the clustering approaches and to find the optimum values for the preclustering parameters is presented here. We begin by reminding of the two parameters for canopy clustering, the T_1 and T_2 values, which are used to form the initial clusters. We begin our experiments by setting both T_1 and T_2 to the same value, and iterate over a set of values. For example, in Cosine distance measure we used values from 0.91 to 0.99

Distance Measure	T_1	T_2	Threshold
Chebyshev	0.4	0.6	-
Cosine	0.95	0.97	0.4
Euclidean	0.85	0.82	-
Manhattan	1.94	1.94	0.2
Minkowski	0.75	0.8	-
Squared Euclidean	0.7	0.7	-
Tanimoto	0.98	0.99	0.1

Table 6.2: Results for optimizing parameters for K-Means clustering. Empty content for threshold indicates that it had no effect on the clustering for that distance measure.

with an increment of 0.01, these values depend on the distance measures used. Once we have found a value that gives good results for our metrics, we set this value to T_2. The parameter T_2 is the more dominant and it defines the size of the clusters by excluding the data points withing its range from other clusters. We then continue by changing only the T_1 value in the neighborhood of T_2 and choose the value which gives the best results in our metrics. Finally, we experiment with the clustering threshold parameter with values from 0.1 to 1.0, and select the best value based on our metrics.

Appendix C and Appendix D presents a sample result of experiments with clustering algorithms, we only present the results for K-Means using Cosine distance and for LDA for brevity. All the other distance measures were experimented with the same methods and produced similar results. We used the optimal values for preprocessing the text that we found in the previous experiments and compared the results of different parameter values for both clustering algorithms.

By inspecting the results of K-Means clustering with different distance measures and values for canopy clustering, we can find the optimum parameter values. There are some values in the results that can be pointed out, for example, the stopping threshold for clustering did not have any effect on some distance measures. With the dataset we used for this experiment we found the optimum parameter around the same number of clusters independent of the distance measure, the number of clusters with the best values for the metrics were around 35 to 40 clusters. The results can be found in more details in Appendix C.

As LDA has only one parameter, the number of topics, it was easier to experiment with. The computational cost of computing the metrics for the results were found to be much higher than for K-Means results, and for this reason we experimented with the number of topics up to 70 with the dataset

of 100 users. As it can be seen in Appendix D, it is not easy to select the optimum number of topics. We chose 40 topics as our parameter as it does provide a local maximum to average inter cluster distances, if we do not consider the results from small number of clusters. Also this value is aligned with the number of clusters we found for the optimal parameters for K-Means clustering. Later in our experiments for other datasets, we found out that 40 topics do not provide good results always, therefore, we chose to use the number of clusters the canopy clustering generated as the number of topics for LDA.

6.4 Recommendations

To evaluate the recommendations we have divided the feeds we used to categories that describe the common topic of the feed. This way we can evaluate whether the recommended feeds are from the relevant categories or not, and we can evaluate the approach based on this. We can compute the accuracy of the recommendation by measuring the ratio of the recommended feeds that are from the correct categories of the feeds that the user has subscribed.

In recommendation accuracy context, the correctly recommended feed is a feed that is from a category that the user has subscribed feeds from. For example, if a user has subscribed feeds from technology and art categories, correctly recommended feed would be from either technology or from art category.

6.4.1 Accuracy

We measured the accuracy of top five recommended feeds of each approach and computed the percentage of correctly recommended feed categories. For each recommendation approach we computed the recommendations for each user and took the average of the correctly recommended feeds. In order to get at least one recommended feed to be correct the accuracy should be 20% or more.

We used two-tailed Wilcoxon signed rank test to evaluate the difference of two approaches. Wilcoxon signed rank test is similar to t-test and measures the differences of the results from two algorithms without any distributional assumptions of the differences between the results of two algorithms [12]. We used the number of correctly recommended feeds as the value to use in the test and compared the best performing approaches against the baseline method.

Our first experiment was with 25 users with 2 random categories for each user. Each user had 5 feeds from the two categories so that each user had 10 feeds in total. The resulting accuracies for this experiment can be seen in figures E.2 and E.1 in the Appendix E. We can observe that the baseline recommender used in the experiments performs well in this dataset achieving a 40% accuracy. By inspecting the baseline accuracy with different feed weighting and distance measures, we can observe that the maximum accuracy is 40% with Cosine distance measure and the lowest accuracy 33.6% with Squared Euclidean and Minkowski distance measures with title similarity.

We can also notice that in our approaches K-Means based recommenders perform well in user based approaches and LDA based recommenders perform better in content based approaches. Collaborative filtering does not perform well in this experiment, as expected, because in randomly generated users we did not input any favourite feeds for users. We can observe the cold start problem for collaborative filtering in these experiments, which occurs when there are not enough favourite values in the system.

Inspecting the results of different weighting approaches for the feeds we can see that none of them improved the accuracy. Inverse update frequency is the most promising of the weighting approaches, but even it does not improve the accuracy significantly. Title similarity weighting using WordNet was the best approach for K-Means and content length weighting was best for LDA. When we compare the results of default K-Means and K-Means with semantically generalized documents, we can observe that the results get worse when using semantically generalized documents. With LDA the results of using semantically generalized documents are similar. To semantically generalize the documents we used 100 most probable words for LDA topic models and 100 terms with highest tf-idf values for K-Means documents.

By performing Wilcoxon signed rank test on the best performing approaches against baseline method we can observe that the results are not statistically significantly better. Table 6.3 shows the p-values for the best performing approaches and we can see that they are not even close to being significantly more accurate at $p \leq 0.05$ level. The approaches that did not perform so well were actually statistically less accurate than the baseline approach at $p \leq 0.05$. We can see that the K-Means content based approaches and the LDA user based approaches are statistically worse than the baseline approach. For the K-Means user based approach and the LDA content based approach there is not that significant difference, so we cannot make any decision on the performance against baseline approach based on these results.

The top-k similarity measures and Hungarian Algorithm are not significantly different of baseline approach in this small dataset. Each of the three

Approach	p-value
K-Means Content Cosine	0.02144
K-Means Content Manhattan	0.07346
K-Means Content Tanimoto	0.1936
K-Means User Cosine	0.7414
K-Means User Manhattan	0.13104
K-Means User Tanimoto	0.5892
LDA Content Cosine	0.28462
LDA Content Manhattan	0.4965
LDA Content Tanimoto	0.28014
LDA User Cosine	0.01596
LDA User Manhattan	0.01684
LDA User Tanimoto	0.01552

Table 6.3: Significance results for 25 user dataset of top-3 K-Means and LDA approaches without weighting.

approaches have a p-value of 0.6672 with Wilcoxon signed rank test.

In the dataset used in the previous experiment the number of users was rather small, and this probably made it easier for our baseline method to perform as well as it did. To be able to decide whether our approach is better than baseline, we experimented with a 50 user dataset where each user had 4 random categories of interest and 2 feeds from each category resulting to a total of 8 feeds per user.

The results, shown in Figure E.5 in Appendix E, for 50 users experiment indicate that the accuracy of baseline approach degrades as more users are added to the system, as expected. The baseline accuracy dropped from 40% to 22.8%, which is nearly a half of the previous accuracy. The average accuracy of the LDA basic approach also dropped slightly from 30.2% to 27.9%, but now some approaches with LDA are clearly better than the baseline approach. Also for the user based recommendations of LDA, the accuracy is increased with more users. For the K-Means based approaches the accuracy improved overall, the maximum accuracy was increased and the average accuracy was 27.9% while it previously was 20.9%. The similarity measures that used top-k similarity measures to find the most similar user did not have a big difference between these two experiments in accuracy.

We can also observe that when we increased the number of users in the system, semantically generalized documents improved the quality of the K-Means user based recommendations. Also with the LDA based approaches we can see that the IUF weighting improves the quality for content based approaches when more users are added. With the LDA content based recom-

Approach	p-value
K-Means Content Cosine	0.4654
K-Means Content Manhattan	0.33204
K-Means Content Tanimoto	0.45326
K-Means User Cosine	0.00128
K-Means User Manhattan	0.00318
K-Means User Tanimoto	0.00032
LDA Content Chebyshev	0.04136
LDA Content Euclidean	0.03156
LDA Content Minkowski	0.0226
LDA User Euclidean	0.41794
LDA User Manhattan	0.84148
LDA User Minkowski	0.35758

Table 6.4: Significance results for 50 user dataset of top-3 K-Means and LDA approaches without weighting.

mendations using weighting of the similarity of the feed titles with WordNet, improves the quality of recommendations. The accuracies of the LDA approaches with WordNet title similarity weighting with Chebyshev, Minkowski and Squared Euclidean distances are statistically significantly better with p values of 0.00044, 0.00108 and 0.0001 respectively.

Appendix E Figure E.3 presents the results for using top-k measures to compute the most similar user. These measures are statistically better than the baseline approaches with p-values of 0.00034, 0.00032 and 0.00032 for Hungarian Algorithm, Top-K Kendall's Tau and Top-K Spearman's Footrule respectively.

By computing statistical tests on the results, presented in Table 6.4, we can observe that the K-Means user based approaches and the LDA content based approaches are statistically better than the baseline method at $p \leq 0.05$ level. We can also observe that the significance of the results are getting better in a sense that the previously statistically less accurate approaches are no longer performing statistically less accurately.

The results of 50 user dataset were still not quite clear whether our approach is an improvement compared to the baseline approach or not, we increased the size of the dataset to 100 users. With this dataset the computation times begun to increase in a single computer environment, and as it is clear that some distance measures perform better than others, we used only a subset of distance measures in our following experiments. We used Cosine, Manhattan and Tanimoto distance measures as they gave the best accuracies on the experiments so far with K-Means. For LDA we first compared the

Approach	p-value
K-Means Content Cosine	0.18684
K-Means Content Manhattan	0.14156
K-Means Content Tanimoto	0.23014
K-Means User Cosine	0
K-Means User Manhattan	0
K-Means User Tanimoto	0
LDA Content Cosine	0.00142
LDA Content Manhattan	0.00374
LDA Content Tanimoto	0.002
LDA User Cosine	0.79486
LDA User Manhattan	0.5157
LDA User Tanimoto	0.53526

Table 6.5: Significance results for 100 user dataset of top-3 K-Means and LDA approaches without weighting.

best performing distance measures and made a decision based on results, to select the three best performing measures. From the results presented in Figure 6.1 we can observe that the Cosine, Manhattan and Tanimoto distance measures perform best with LDA also, when compared against Euclidean and Minkowski. Because canopy clustering created 78 clusters in this dataset, we decided to use also the same number of topics for LDA.

Inspecting the results of 100 user dataset, presented in Figure 6.1, we can observe that the K-Means user based recommendation outperforms the baseline and the K-Means content based recommendations in accuracy. K-Means has the best accuracy of 47.4% in the experiments so far, with Cosine and Tanimoto distance measures in user based approach. The same behavior was present in the smaller datasets similarly. For LDA the recommendation approaches perform in the opposite way so that the content based approaches outperform the user based approaches. We can also observe that the title similarity using WordNet improves the accuracy of the methods, while other weighting approaches do not have a significant positive effect. The best accuracy for LDA was with WordNet title similarity and the Cosine distance measure in the content based approach with an accuracy of 38%. Also with the top-k based similarity measures, presented in Figure 6.2, the best performance is achieved with the biggest dataset with an accuracy of 47.2%. The significance of these resulting accuracies can be seen in Table 6.5, where we can see that the aforementioned approaches are statistically significantly better than the baseline approach.

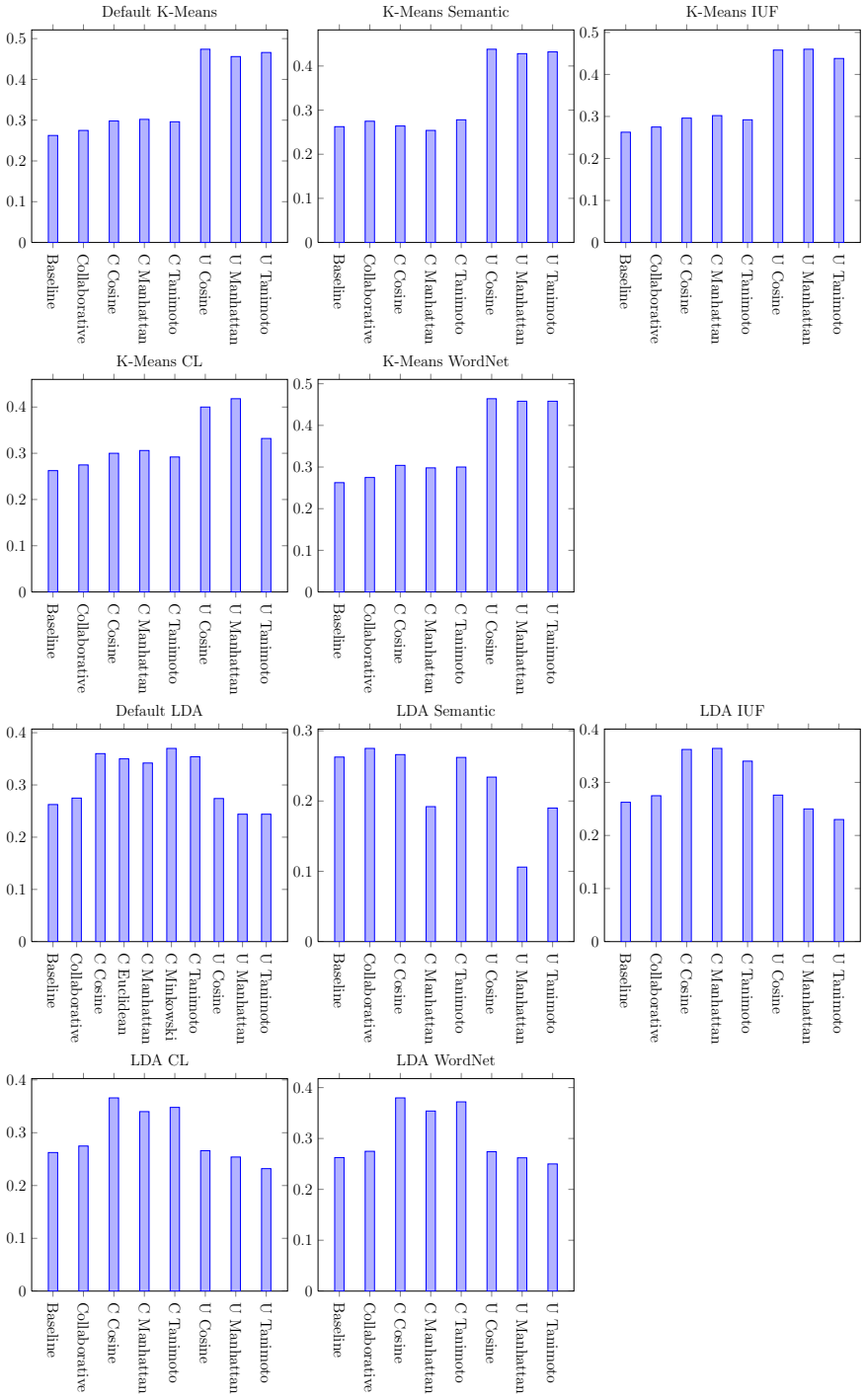


Figure 6.1: Accuracy results of 100 user dataset for K-Means and LDA. K-Means results are the 5 charts above and LDA are the 5 charts below. C and U denote the content and user based approaches respectively.

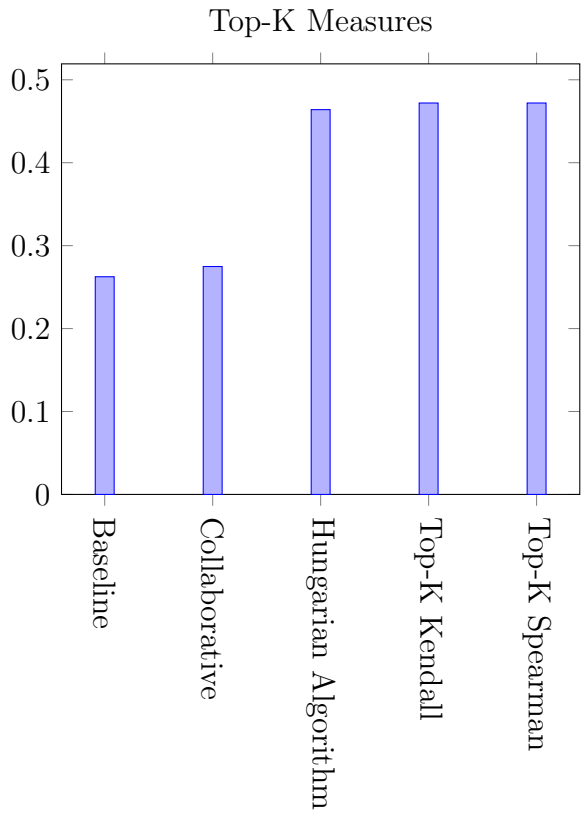


Figure 6.2: Accuracies for top-k recommendation in a 100 user dataset.

6.4.2 Simulating Real Users

Where the previous experiments were conducted in an automated way to compare the correctness of the recommendation in order to allow much bigger user population, the next experiments were conducted manually in a smaller set of users. This allows us to eliminate the possibility of a wrongly categorized feed by manually comparing the recommended feed to the feeds the user is already interested of. Because of the prototype level of our application we were not able to provide internet access to our application or neither package the application to a distributable format. We performed our experiments with real human-like users in a local environment, where the results were examined by hand. For this experiment we created 12 users with 4 feeds from 3 different interest categories and examined the recommendation results by hand in an objective manner.

Listing in Appendix F shows an example of recommendation results with Cosine distance measure from the dataset of hundred users. In the listing

there are listed the original user feeds and recommendations from different approaches. Each feed is identified by the category it belongs to and the title of the feed. From these results we can check the results automatically by checking the number of feeds from the correct categories, in this case from art, misc, science and technology and business. In the manual evaluation we can check the title of the recommended feed, or if the title is not distinctive enough, we can check the actual content of the feed. For this example user, the LDA content based approach gives the best recommendations, then the K-Means content based and the LDA user based approaches. The K-Means user based approach performs the worst of the four basic approaches, even though the average accuracy of the K-Means user based approach was the best. The top-k measures recommend as good results as the K-Means content and the LDA user based approaches, while Hungarian Algorithm gives slightly worse results than the top-k.

In the small scale experiment of the accuracy with a human valued recommendation, we used the best performing approach for both K-Means and LDA and both baseline and collaborative filtering. To approve the recommended feed as correct we inspected the feeds the user had subscribed and compared if the recommended feeds were of a similar field. For example, if a user had art and business feeds in recommendations, we inspected the contents of the subscribed feeds if they were from either of these fields.

For the K-Means user based approach with Cosine similarity the accuracy was 78.3% and with the LDA content based approach with Cosine similarity the accuracy was 58.3%. With the baseline approach the accuracy was 30%. We also experimented the effect of adding favourite feeds which yielded a 38.3% accuracy to collaborative filtering. Feed favouring increased the accuracy of the baseline approach to 31.7%. The K-Means and LDA the accuracies decreased to 63.3% and 51.7% respectively.

To compare the accuracies against the automatic accuracies, presented earlier, we ran automated accuracies also. For the K-Means results the automated accuracy was 45% and for LDA 56.7%. The baseline accuracy was in automated calculations 48.3% and collaborative was 33.3% accurate. When we compare these to the results achieved by human valuation, the results are quite similar with LDA results. For K-Means the accuracy is significantly smaller with automated computations than with human valuations. The baseline approach achieved higher accuracy using automated accuracy computation.

6.5 Distance Measures

We can use the results of the recommendation experiments to evaluate the performance of the different vector space distance measures described in subsection 3.5.1. In the book by Weiss et al. [47] it is mentioned that cosine similarity is the classical approach to measure the similarity of documents. Tanimoto distance measure being similar to cosine distance, we can expect also good results from it.

Based on the results of the experiments with 50 users, results shown in Figure E.5, for K-Means we can see that the Cosine, Manhattan and Tanimoto distance measures give the best results. For LDA results, picking the best performing distance measure was not that easy task, due to the results being more similar between the distance measures. With different weighting approaches we can observe that Cosine, Manhattan and Tanimoto distance measures still perform well. Interestingly also Euclidean, Minkowski and Squared Euclidean distance measures perform well in some weighting approaches with LDA.

We had three different approaches to measure the user similarities, the Hungarian algorithm and the two top-k similarity measures. By observing the results of them in Figure E.3 and Figure 6.2 we can notice that they all perform similarly in terms of accuracy. The bigger the dataset the more accurately they retrieved the closest user, and thus providing more accurate recommendations.

We also experimented with the effect of using different WordNet similarity algorithms, and discovered that in our small scale measures of measuring the title similarity, the choice of algorithm did not have a significant difference in the distance. Some approaches were much more computationally expensive than others. Therefore, we used the approach that was feasible to compute and provided good results, this was the Resnik approach to measure word similarities.

For the top-k distance measures of Spearman's Footrule and Kendall's Tau, we experimented with the number of top words needed from the topics. Spearman's Footrule needed 250 top words in topics to perform well, increasing or decreasing this value did not improve the results. Kendall's Tau distance needed only top 50 words from the topics to perform well, and similarly changing this number did not have a significant improvement to the results.

6.6 User Distance Measures

We can evaluate the user distance measures by computing user to user distances. We experimented in a 100 user dataset where we had 5 different user groups, so we should be able to see 5 groups in the user distance heat maps. Figure 6.3 through Figure 6.13 presents the heat maps of the user to user distances of different distance measures. We present only the top three best performing distances as other distance measures did not produce as good clusters of the users to the expected 5 groups.

We can see that top-k based measures, Spearman's footrule and Kendall's Tau, are good measures to user to user distance computations. Hungarian algorithm also provides good results but slightly worse than the previously mentioned approaches. LDA does not provide as good results as the others and there are no clear groups of users visible.

The results provide an explanation to the results of recommendation accuracy, where the K-Means user based approach provided good results with the Hungarian algorithm and the top-k similarities. LDA on the other hand had better results in the content based approaches, which is probably due to the inferior performance in the user distance computations.

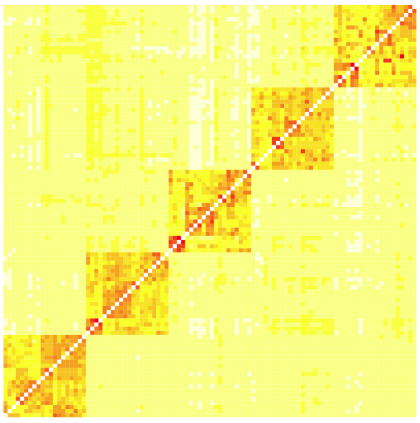


Figure 6.3: Spearman's Footrule Distance

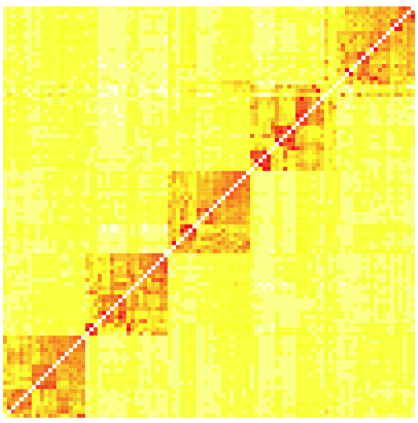


Figure 6.4: Kendall's Tau Distance

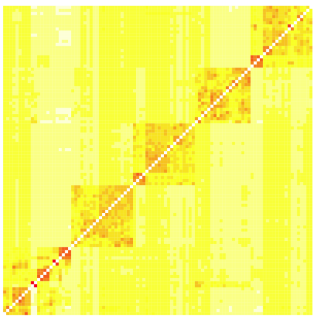


Figure 6.5: K-Means Cosine Distance

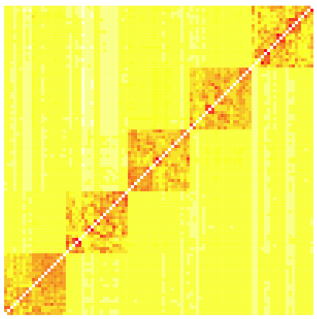


Figure 6.6: K-Means Manhattan Distance

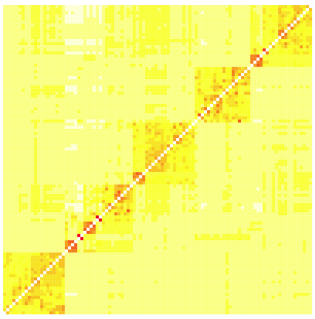


Figure 6.7: K-Means Tanimoto Distance

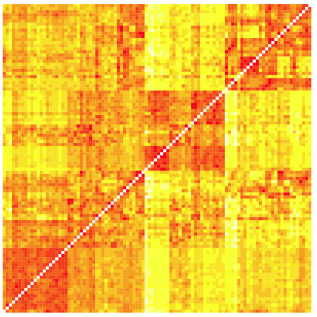


Figure 6.8: LDA Cosine Distance

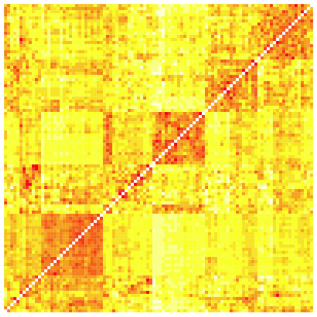


Figure 6.9: LDA Manhattan Distance

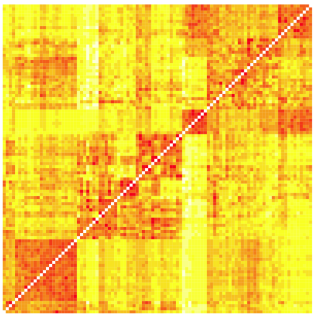


Figure 6.10: LDA Tanimoto Distance

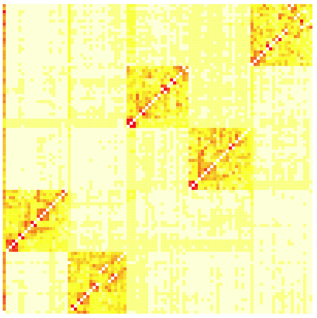


Figure 6.11: Hungarian Algorithm Cosine Distance

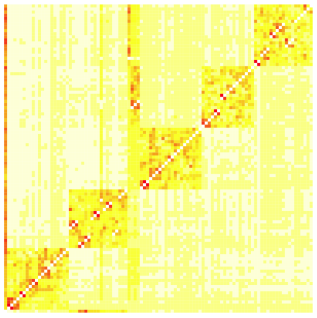


Figure 6.12: Hungarian Algorithm Manhattan Distance

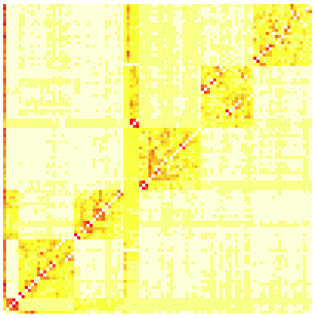


Figure 6.13: Hungarian Algorithm Tanimoto Distance

Chapter 7

Evaluation of Results

We presented different approaches to RSS recommendation systems in chapter 2 and we can use the results of those approaches to evaluate the results of our approach. For some of the approaches there is no information available about the accuracy of the recommendations. Also the measure of accuracy differs in the approaches, for example, in paper by Zhao et al. [50] the authors use the number of favourite feeds recommended as a measure of accuracy.

7.1 Comparing Results

Zhao et al. [50] presented in their paper only the precise accuracy for the combinations of similarity measures. The single measure accuracy is only shown in a bar chart, from which we can estimate the accuracy of using a single feature. The accuracy of a single feature ranges from 2.5% to 20%, while by combining the measures the accuracy ranges from 10% to 80%. The difference to our experiments is that Zhao et al. use at minimum top-10 feeds to recommend, while we used top-5.

If we compare the results of single feature similarity accuracies we can observe that our approach has the best accuracy of 47.4% where as Zhao et al. reached 20% accuracy, our worst accuracy was 6.4% with a single feature. Based on the accuracies, our approach reached better performance on single feature, however, when multiple features are used Zhao et al. approach is significantly better with 80% accuracy. When we compare the results of our manual experiment, we can reach a similar level of average accuracy with 78.3%.

NectaRSS, introduced in chapter 2, also achieves over 80% accuracy in their experiments presented in paper by Samper et al. [43]. NectaRSS' approach is quite different from ours and it uses an incremental approach

to learn user preferences, NectaRSS also uses the title similarity of the feeds with a different approach from ours. NectaRSS uses the vector space distance measure for the title similarity, where we used the WordNet based similarity.

Both of the approaches, we compared our application to, used a rather small user base to perform the evaluation. Zhao et al. used 20 volunteers and NectaRSS used 15 volunteers and 30 sessions, in our experiment which simulated user behavior there were 12 generated users, which is in the same range as the other approaches. This can give some indication that our approach can perform at the same level of accuracy as the compared approaches.

To be able to make reasonable conclusions on the quality of our approach compared to the other approaches, the dataset should be the same. We can observe that the accuracy of using human evaluation is much better than the automated approach. Also if we compare the accuracies of our automated experiments, we can see that the accuracy improves as the number of users increases, which can give some indication that the document clustering approach requires more users to be able to add significant improvement.

7.2 Observations From Results

When comparing the two different approaches to experiment, we can observe that the accuracy of K-Means approach is lower in automated experiments. This gives us a reason to suspect that should the experiments with bigger dataset be evaluated by humans, the accuracies could have been much better. For LDA the difference is not that big between these two evaluation methods, so we suspect that those approaches do not reach the accuracy of K-Means.

By inspecting the results of the different weighting approaches, for both modeling the quality of the feed as well as the similarity of the titles, we can observe that none of the weighting approaches had any significant effect. WordNet based title similarity had the best effect on the accuracy with LDA. Generating semantically generalized documents or semantically generalized topics did not help to increase the accuracy of recommendations. However, both of these WordNet based approaches gave the most promising results among the weighting approaches we evaluated. Both methods to model the feed quality, inverse update frequency and content length, appear to only worsen the results with some distance measures and have little to no effect on other distance measures.

The results of user distance measure comparisons provide a good overview of the overall quality of the approach. We can see that the K-Means based approaches provide much clearer clusters of users, where the LDA based approaches generate much more noise between clusters of users.

Chapter 8

Discussion and Conclusions

In this thesis, we have developed a prototype application to recommend RSS feeds based on the user's preferences derived from the feed subscriptions. The main goal was to further develop the current state-of-the-art methods in feed recommendations. To our best knowledge, this is the first time WordNet based approaches have been applied to this kind of a problem.

Our first goal was to investigate the current trends in document clustering and choose the appropriate approaches from them. We selected K-Means and Latent Dirichlet Allocation as our document clustering techniques based on the popularity and the performance in the domain. In addition to these two approaches we chose canopy clustering as the preclustering technique to overcome the difficult decision of choosing the correct number of clusters. For canopy clustering we could perform parameter optimization to our seven distance measures using clustering quality metrics. We discovered that by optimizing the parameters for canopy clustering we can obtain meaningful clusters on multiple datasets and maintain high quality of clustering metrics. For Latent Dirichlet Allocation we can optimize the number of topics using the same clustering quality metrics. We discovered later in our experiments that the optimized number of clusters discovered earlier to our problem was not good enough for other datasets. To overcome this issue we discovered that using the number of clusters generated by canopy clustering, turned out to be a good solution. Canopy clustering can then be used as an unsupervised preclustering technique for both of our document clustering approaches.

The second goal was to analyze the previously clustered documents and generate meaningful user models based on the results. We exploited the document similarities based on vector space distance measures to generate a model of the user's interests. We discovered that some distance measures perform better than others to cluster the user models, namely Cosine, Tanimoto and Manhattan distances performed well. We also noticed a difference

in the user model quality between the document clustering approaches, for Latent Dirichlet Allocation we used the topic models of the user's feeds as a user model, which turned out to be of lesser quality than the K-Means approach. The K-Means approach used the feeds tf-idf representations to model the feed, from which a better user model was retrieved.

Our third goal was to recommend the most similar feeds based on the user models generated. We used two approaches to compute the recommendations, one being based purely on the contents of the feeds and other based on the user models. We discovered that using state-of-the-art methods in feed recommendations did not help in our approaches significantly. We developed a novel methods based on shallow word ontologies to improve the similarity measures. We discovered that converting the terms in a feed to a more general terms using WordNet did not improve neither K-Means or LDA based recommendations. However, when we used WordNet to measure the similarity of the feed titles, we did observe an improvement to the quality of the LDA based recommendations.

The final goal was to combine all the methods we have developed to a functional application. We leveraged the available software libraries from machine learning in order to be able to focus on developing new methods, rather than implement everything ourselves from the beginning. By using widely used libraries we can, for example, improve the scalability of our application and leverage the future updates to algorithms in order to further improve the accuracy.

We conclude this thesis with some directions for future work. To begin with, we experimented with only two document clustering approaches, but we did implement Suffix-Tree Clustering (STC) up to the clustering phase. We did not implement STC based user models or recommendations, but based on the quality of the clustering resulting from STC we can expect that the results should be comparable. Another improvement to document clustering can be achieved by exploiting soft cluster association provided by LDA topic models. We only included the most probable topic in our approach and including more than one topic for a feed could increase the performance of LDA recommendations. Our application does not, at its current state, allow for a wider usage than experimental, an improvement to this can be achieved by packaging it to an internet accessible application. In addition, we have so far implemented scalable algorithms in the document clustering part, but we believe that also the rest of our application can be made scalable. To improve the scalability we have given thought for it by making our approaches modular and with that it should be possible to convert them, for instance, to map-reduce jobs. Exploiting more the information WordNet provides, or any other word ontology, would be an interesting future direction.

Bibliography

[1] Apache. Apache hadoop, May 2013. URL http://hadoop.apache.org/.

[2] Apache. Apache lucene, May 2013. URL http://lucene.apache.org/.

[3] Apache. Apache wink, May 2013. URL http://wink.apache.org/.

[4] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 805–810. LAWRENCE ERLBAUM ASSOCIATES LTD, 2003.

[5] Pierpaolo Basile, Marco de Gemmis, Anna Lisa Gentile, Pasquale Lops, and Giovanni Semeraro. Uniba: Jigsaw algorithm for word sense disambiguation. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 398–401, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/S/S07/S07-1088.

[6] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[7] David M Blei and J Lafferty. Topic models. *Text mining: classification, clustering, and applications*, 10:71, 2009.

[8] The NewsBlur Blog. Explaining intelligence, April 2011. URL http://blog.newsblur.com/post/4262089560/explaining-intelligence.

[9] The NewsBlur Blog. Three months to scale newsblur, March 2013. URL http://blog.newsblur.com/post/45632737156/three-months-to-scale-newsblur.

[10] Alexander Budanitsky. Lexical semantic relatedness and its application in natural language processing. 1999.

[11] Ting Chen, Wei-Li Han, Hai-Dong Wang, Yi-xun Zhou, Bin Xu, and Bin-yu Zang. Content recommendation system based on private dynamic user profile. In *Machine Learning and Cybernetics, 2007 International Conference on*, volume 4, pages 2112–2118. IEEE, 2007.

[12] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[13] Persi Diaconis and Ronald L Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 262–268, 1977.

[14] Angela Edmunds and Anne Morris. The problem of information overload in business organisations: a review of the literature. *International journal of information management*, 20(1):17–28, 2000.

[15] Ronald Fagin, Ravi Kumar, and D Sivakumar. Comparing top k lists. *SIAM Journal on Discrete Mathematics*, 17(1):134–160, 2003.

[16] Feedage.com. Rss feed directory - feedage.com, July 2013. URL `http://www.feedage.com`.

[17] Feedly. Building feedly blog, announcing the new feedly mobile, April 2013. URL `http://blog.feedly.com/2013/04/02/announcing-the-new-feedly-mobile-and-welcoming-3-million-reader-refugees/`.

[18] Feeds2Read.net. Feeds2read.net your rss feed for today, July 2013. URL `http://www.feeds2read.net`.

[19] Google. Google reader website, May 2013. URL `http://www.google.com/reader`.

[20] Google. Official blog : A second spring of cleaning, March 2013. URL `http://googleblog.blogspot.com/2013/03/a-second-spring-of-cleaning.html`.

[21] Maria Halkidi and Michalis Vazirgiannis. Clustering validity assessment using multi representatives. In *Proceedings of SETN Conference, Thessaloniki, Greece*, 2002.

[22] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[23] Jonathan Hedley. Jsoup - java html parser, May 2013. URL `http://jsoup.org/`.

[24] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[25] Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database*, 305:305–332, 1998.

[26] Cansheng Ji and Jingyu Zhou. A study on recommendation features for an rss reader. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2010 International Conference on*, pages 193–198. IEEE, 2010.

[27] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[28] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[29] HW Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52(1):7–21, 2005.

[30] Claudia Leacock, George A Miller, and Martin Chodorow. Using corpus statistics and wordnet relations for sense identification. *Computational Linguistics*, 24(1):147–165, 1998.

[31] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM, 1986.

[32] Dekang Lin. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 64–71. Association for Computational Linguistics, 1997.

[33] Stuart Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[34] Andrew McCallum, Kamal Nigam, and Lyle H Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM, 2000.

[35] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[36] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[37] Fionn Murtagh. Clustering in massive data sets. *Handbook of massive data sets*, pages 501–543, 2002.

[38] Adrian P O'Riordan and M Oliver O'Mahony. Intersynd: a web syndication intermediary that makes recommendations. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 299–304. ACM, 2008.

[39] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in action*. Manning, 2011.

[40] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[41] Francesco Ricci and Bracha Shapira. *Recommender systems handbook*. Springer, 2011.

[42] RSS Advisory Board. Rss 2.0 specification, March 30 2009. WWW page of the RSS Advisory Board: `http://www.rssboard.org/rss-specification`. Accessed 11 Apr 2013.

[43] Juan J Samper, Pedro A Castillo, Lourdes Araujo, JJ Merelo, Oscar Cordon, and Fernando Tricas. Nectarss, an intelligent rss feed reader. *Journal of Network and Computer Applications*, 31(4):793–806, 2008.

[44] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[45] Pierre Tirilly, Vincent Claveau, and Patrick Gros. Distances and weighting schemes for bag of visual words image retrieval. In *Proceedings of the international conference on multimedia information retrieval*, pages 323–332. ACM, 2010.

[46] Princeton University. Princeton university "about wordnet.", 2010. URL `http://wordnet.princeton.edu`.

[47] Sholom M Weiss, Nitin Indurkhya, and Tong Zhang. *Fundamentals of predictive text mining*, volume 41. Springer, 2010.

[48] Built With. Web techonology usage trends, July 2013. URL `http://trends.builtwith.com/`.

[49] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[50] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524. ACM, 2002.

[51] Ying Zhao, George Karypis, and Usama Fayyad. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168, 2005.

Appendix A

List of Feed Categories

- antiques
- architecture
- art
- art history
- arts and craft
- blogs
- blogs news
- business
- computers
- consumer electronics
- cooking
- economics
- education
- entertainment
- finance
- food
- games
- health
- health diet
- health fitness
- hobbies
- home
- journals
- literature
- machine learning and data mining
- miscallaneous
- movies
- msn
- news
- photography
- programming
- science
- society
- sports
- travel

Appendix B

Text Preprocessing Results

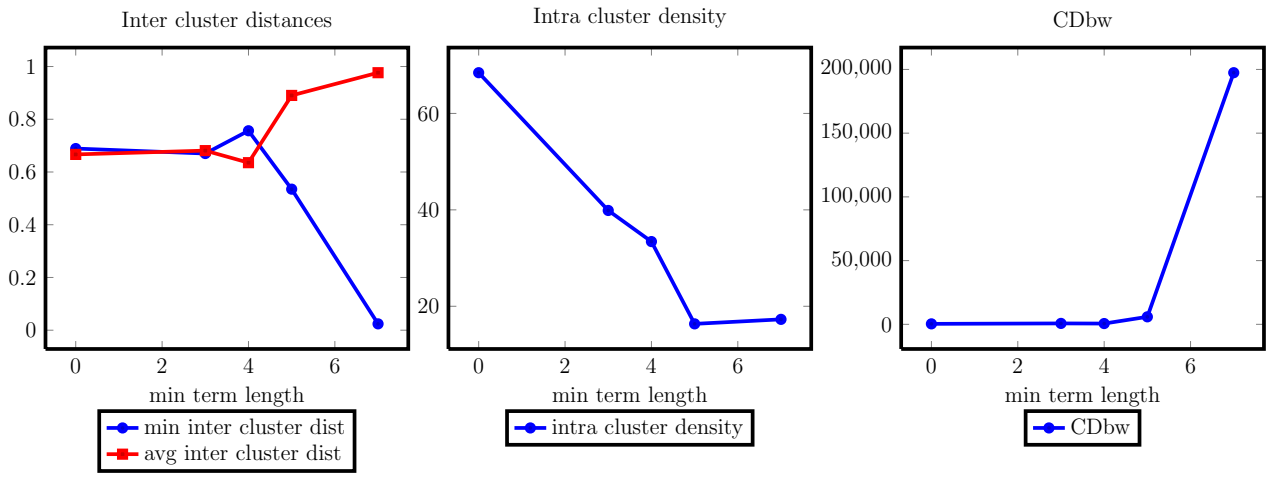


Figure B.1: Effect of preprocessing minimum term length to clustering.

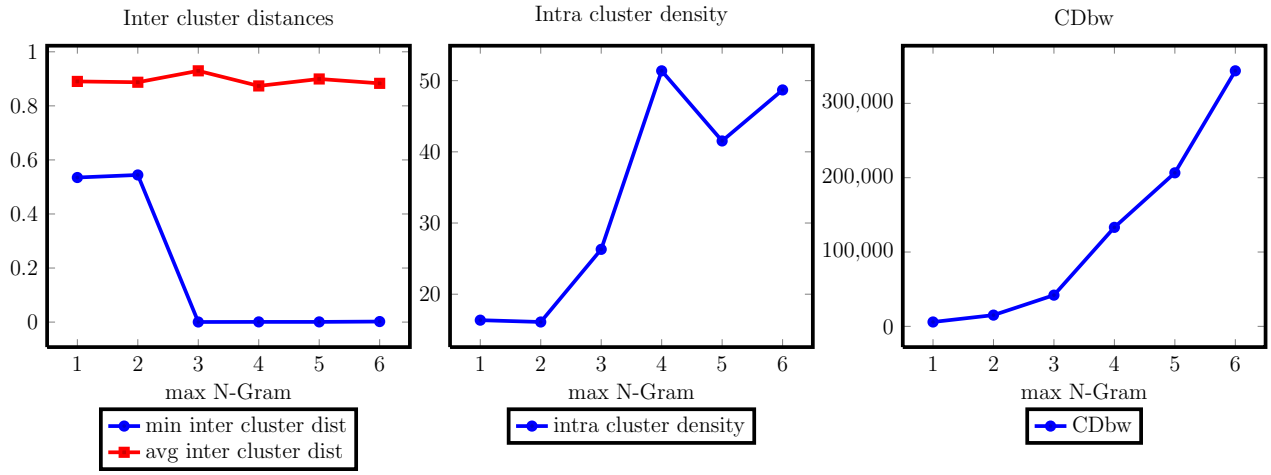


Figure B.2: Effect of preprocessing maximum allowed N-grams to clustering.

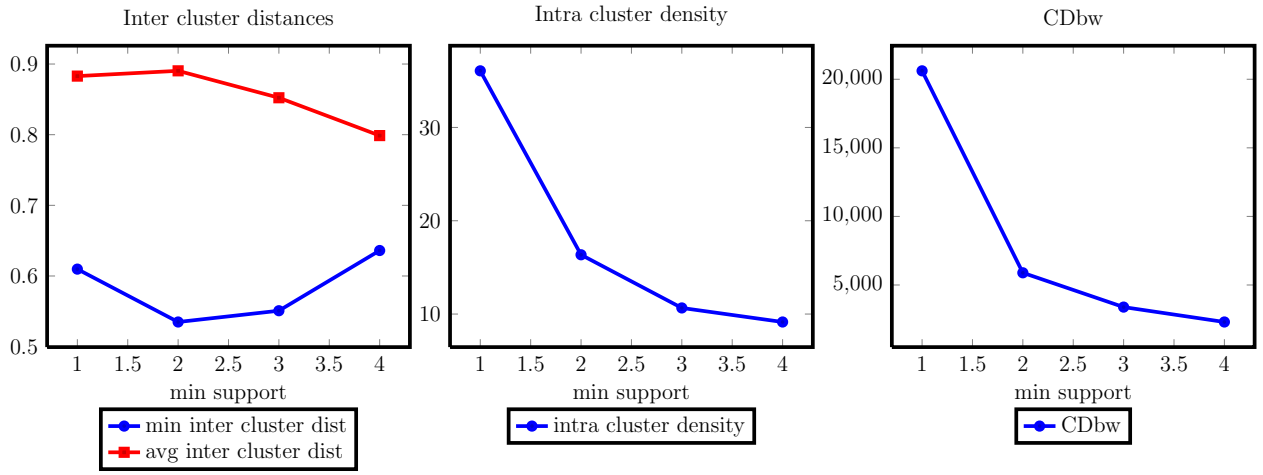


Figure B.3: Effect of preprocessing minimum support for term to clustering.

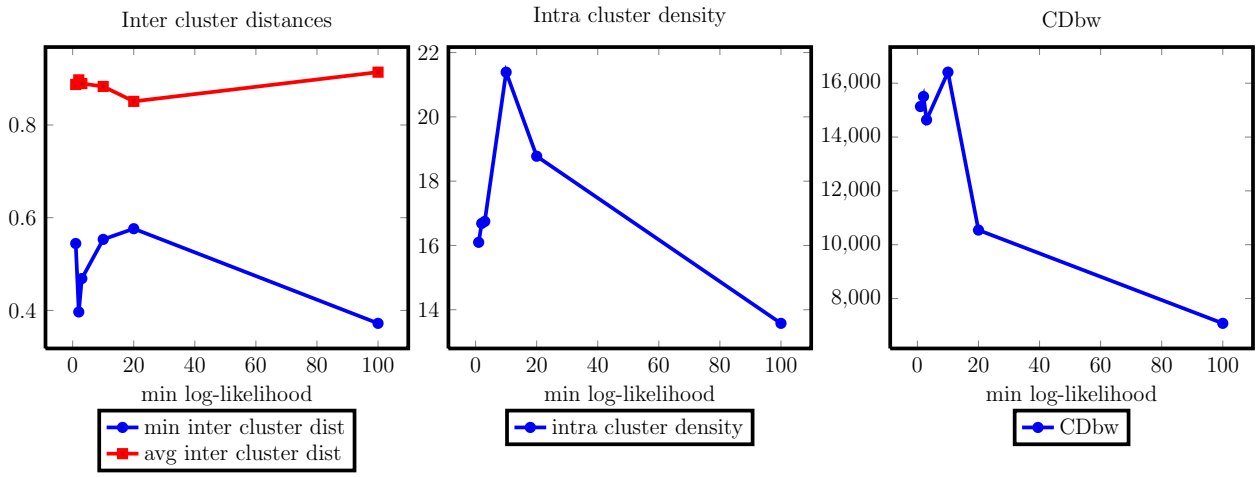


Figure B.4: Effect of preprocessing Minimum Log-Likelihood ration for collocations to clustering.

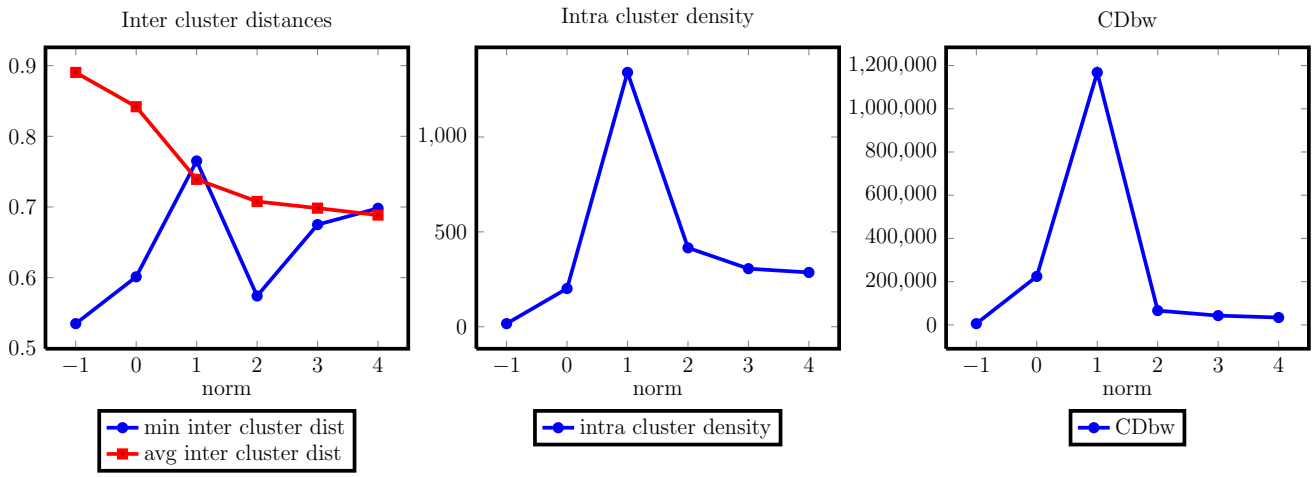


Figure B.5: Effect of preprocessing norm to use to clustering.

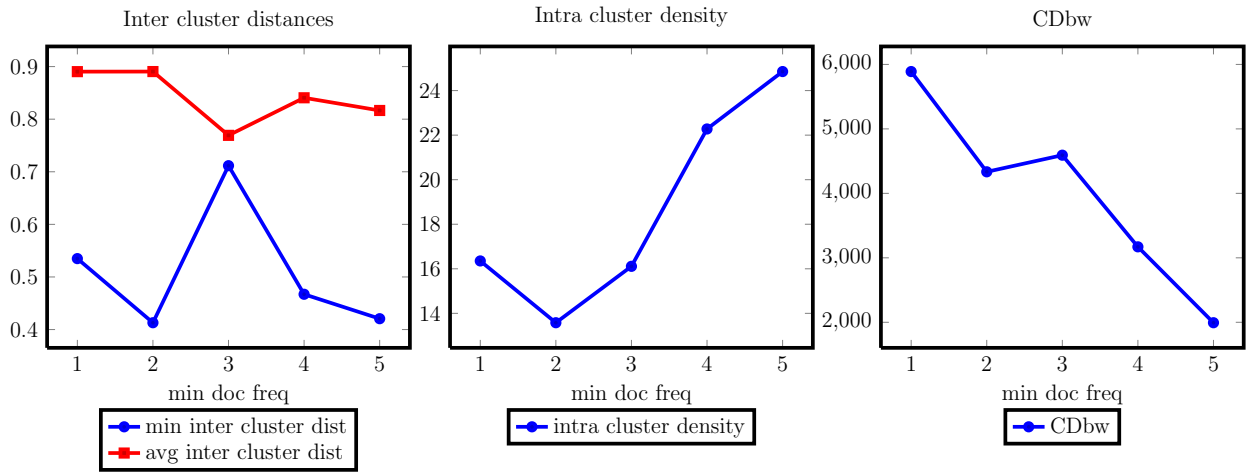
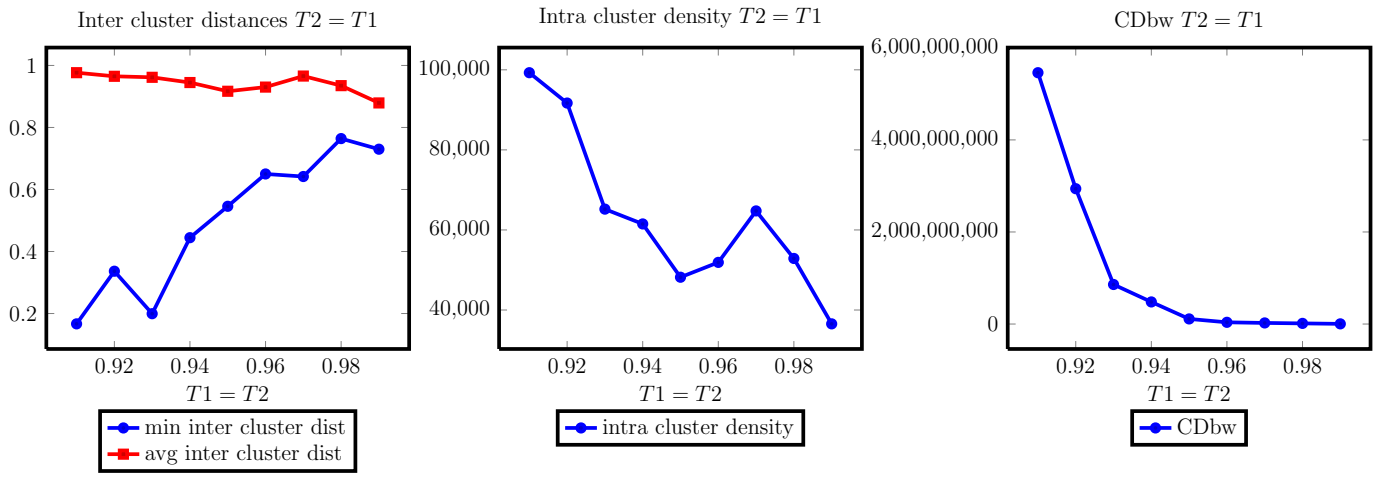


Figure B.6: Effect of preprocessing minimum document frequency to clustering.

Appendix C

K-Means Clustering Results



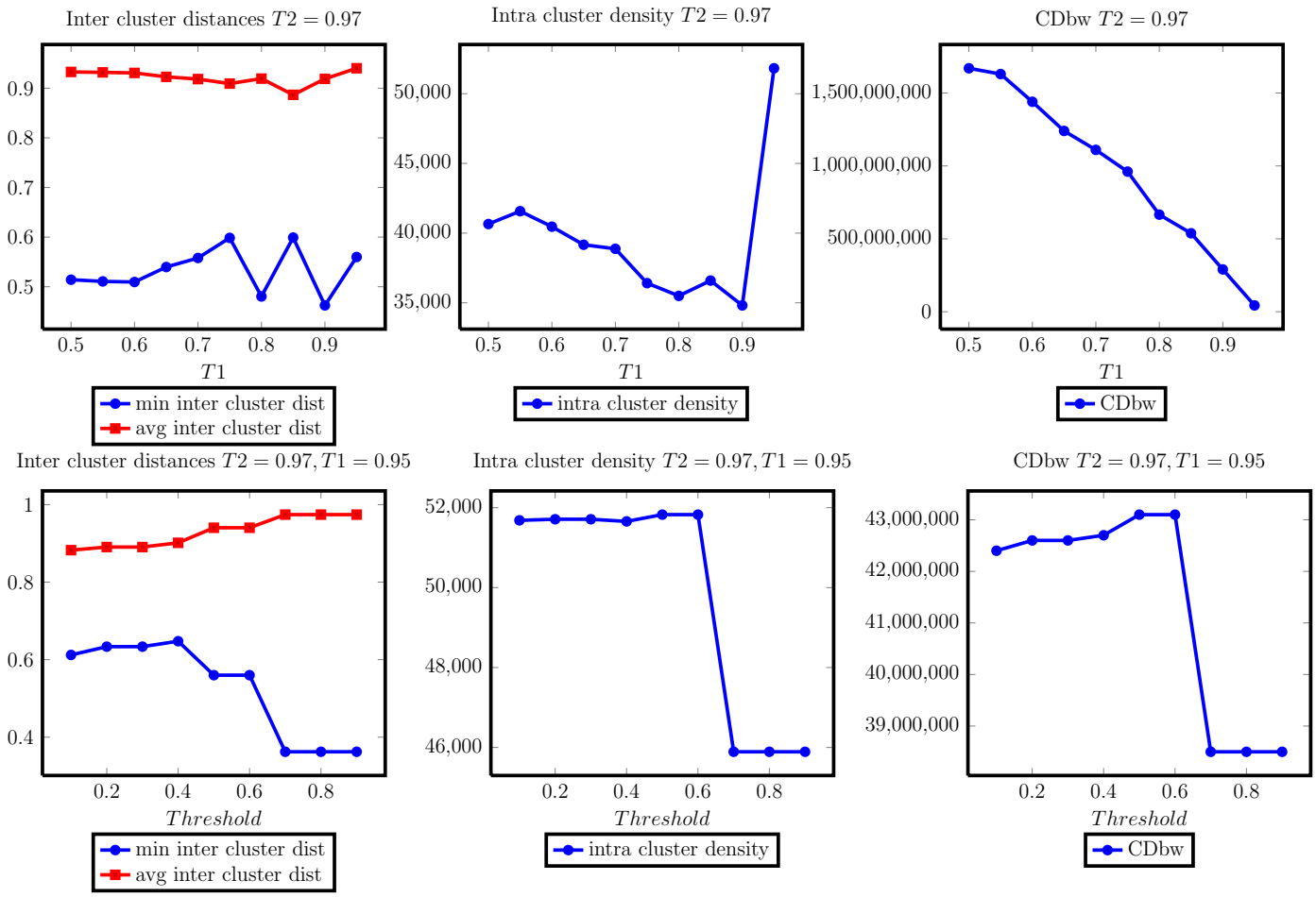


Figure C.1: Results for selecting the optimum values for canopy clustering for cosine K-Means.

Appendix D

LDA Clustering Results

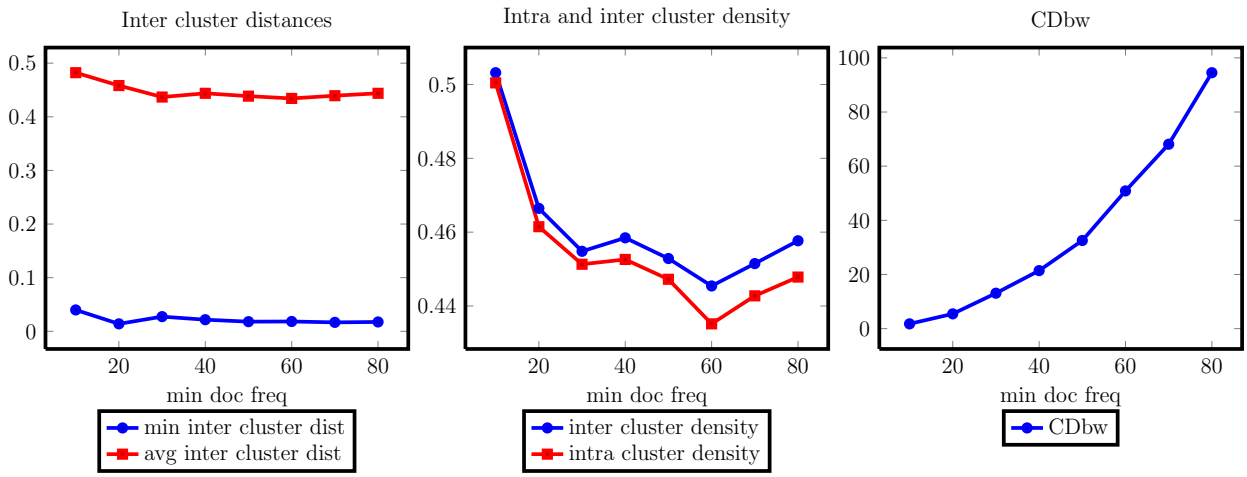


Figure D.1: Results for comparing different number of topics for LDA.

Appendix E

Recommendation Results

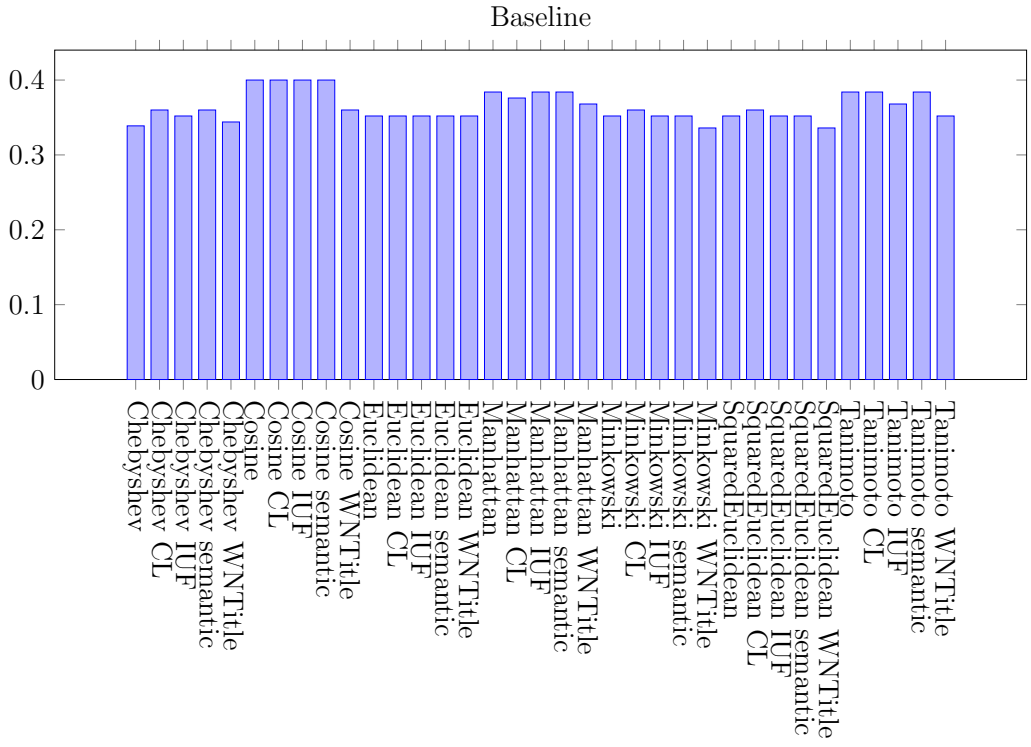


Figure E.1: Accuracies for baseline recommendation in a 25 users dataset with different distance measures.

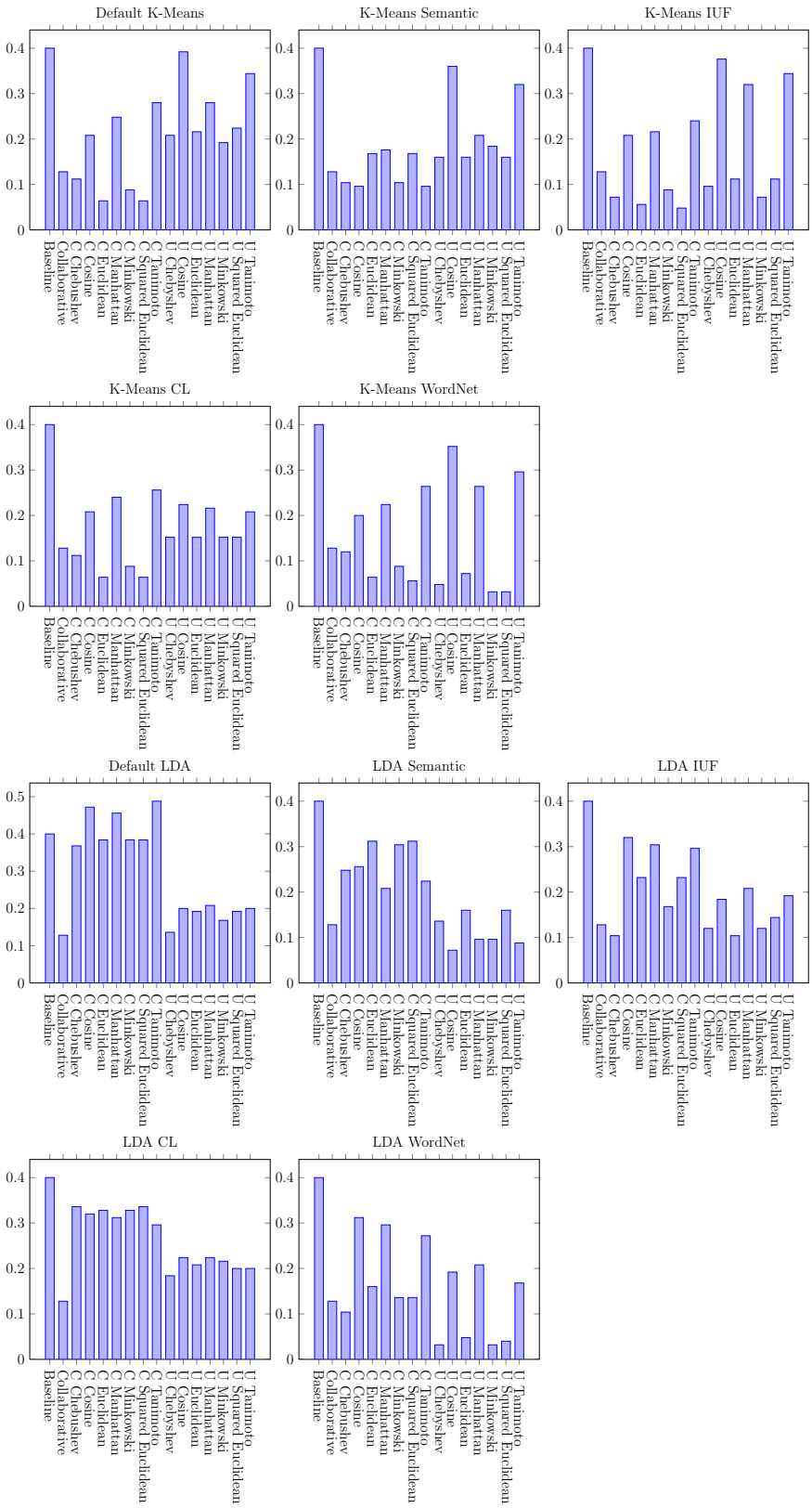


Figure E.2: Accuracy results of 25 user dataset for K-Means and LDA. K-Means results are the 5 charts above and LDA are the 5 charts below. C and U denote the content and user based approaches respectively.

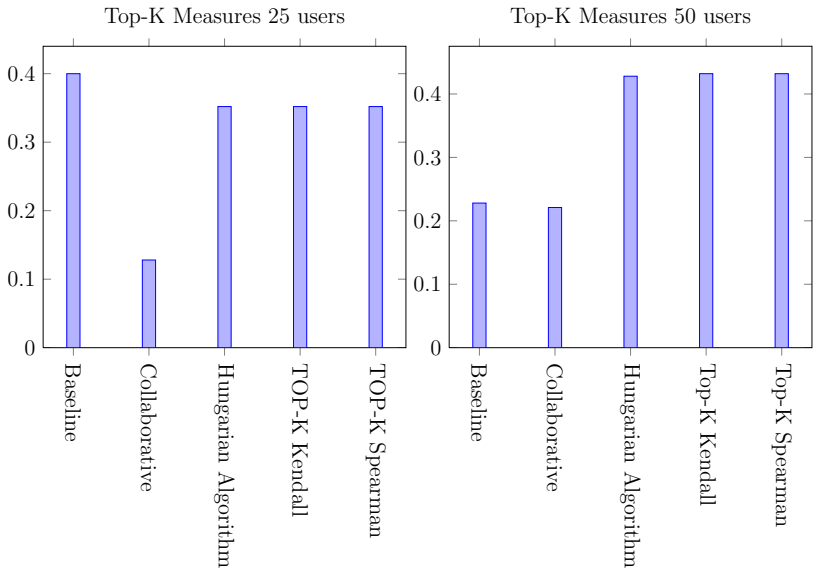


Figure E.3: Accuracies for top-k recommendation in 25 and 50 user datasets.

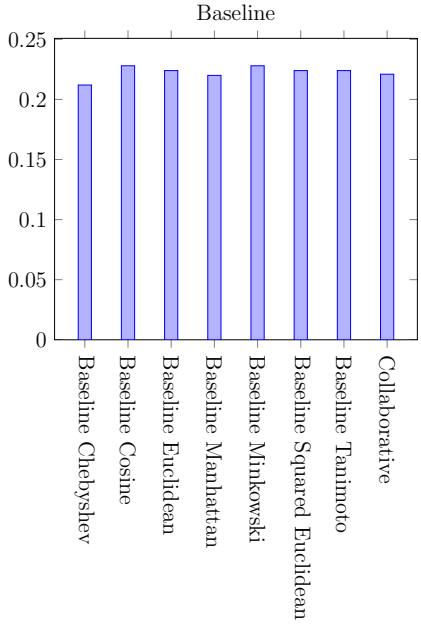


Figure E.4: Accuracies for baseline recommendation in a 50 user dataset with different distance measures.

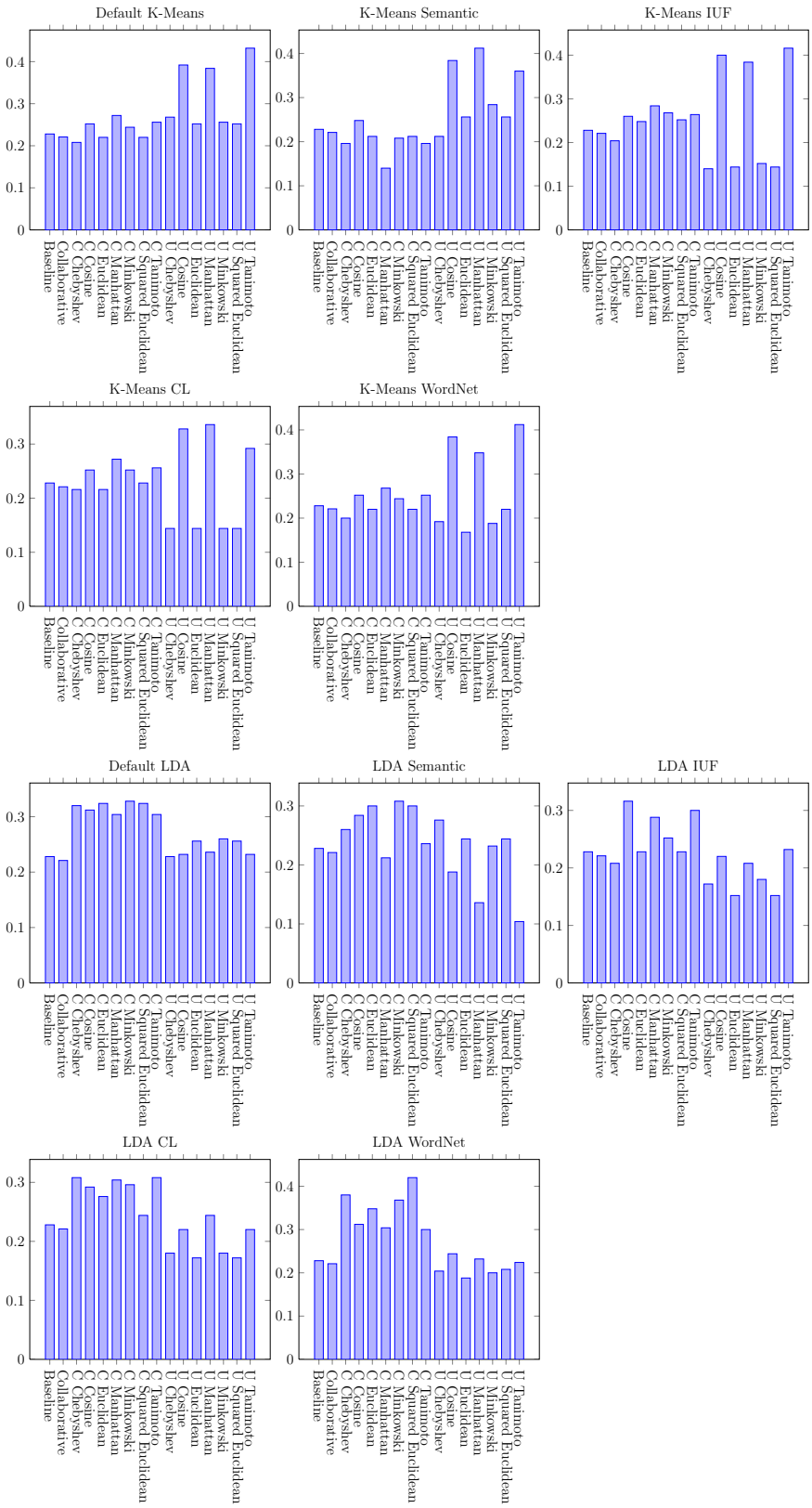


Figure E.5: Accuracy results of 50 user dataset for K-Means and LDA. K-Means results are the 5 charts above and LDA are the 5 charts below. C and U denote the content and user based approaches respectively.

Appendix F

Example Recommended Feeds

```
User Feeds :
  ART : NYT Theater
  ART : MIT News Topic Music technology
  MISC : Feedzilla Quote Of The Day News
  MISC : News
  SCIENCE AND TECHNOLOGY : The Official Google Blog
  BUSINESS : Business and financial news CNN Money.com

Recommended by Baseline
  ART : Wash Post Books
  HOBBIES : Events
  HOME : Wash Post Home Improvement
  MSN : Top 5 Most Recent MSN Articles
  HOBBIES : Travel - travel guides ...

Recommended by Collaborative
  MISC : Feedzilla Australia News
  HOME : Hollis ramblings
  CONSUMER ELECTRONICS : Best Digital Camcorder Reviews
  ART : NYT Arts
  BUSINESS : Businesses For Sale by Owner in Canada

Recommended by K-Means Content with Cosine Distance
  SCIENCE AND TECHNOLOGY : Science
  MISC : Feedzilla Comics News
  CONSUMER ELECTRONICS : Memory Cards Camera Mobile Phone ...
  BUSINESS : Business chicagotribune.com
  HEALTH : Fitness and Nutrition Center
```

```
Recommended by K-Means User with Cosine Distance
  SCIENCE AND TECHNOLOGY : Technology Articles
  ART : MIT News Topic MediaLab
  COMPUTERS : Gadget Advisor
  COMPUTERS : Geeking with Greg
  GAMES : TECHNOCRAZY

Recommended by LDA Content with Cosine Distance
  MISC : Feedzilla Dance News
  SOCIETY : The Society of Qualified Archivists
  MISC : Feedzilla Albany News
  SCIENCE AND TECHNOLOGY : MIT News Topic Mechanical engineering
  ART : MIT News Topic MediaLab

Recommended by LDA User with Cosine Distance
  MISC : Feedzilla Dance News
  SOCIETY : The Society of Qualified Archivists
  CONSUMER ELECTRONICS : Memory Cards Camera Mobile Phone..
  MISC : Feedzilla Albany News
  SCIENCE AND TECHNOLOGY : MIT News Topic Mechanical engineering

Recommended by Hungarian Algorithm with Cosine Distance
  SCIENCE AND TECHNOLOGY : Technology Articles
  ART : MIT News Topic MediaLab
  COMPUTERS : Gadget Advisor
  COMPUTERS : Geeking with Greg
  GAMES : TECHNOCRAZY

Recommended by Kendall's Tau
  MISC : Feedzilla Comics News
  SCIENCE AND TECHNOLOGY : Technology Articles
  ART : MIT News Topic MediaLab
  COMPUTERS : Gadget Advisor
  COMPUTERS : Geeking with Greg
Recommended by Spearman's Footrule
  MISC : Feedzilla Comics News
  SCIENCE AND TECHNOLOGY : Technology Articles
  ART : MIT News Topic MediaLab
  COMPUTERS : Gadget Advisor
  COMPUTERS : Geeking with Greg
```