Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

New ways of browsing and reading news

Master Thesis
February 2012 – August 2012

Author: Dany Sünnen, dany@student.ethz.ch

ETH Zurich - Distributed Computing Group
Supervisors: Mihai Calin and Samuel Welten
Professor Dr. Roger Wattenhofer

Abstract

Nowadays people mainly read news in newspapers, on the Internet and on their mobile device. In recent years, due to the proliferation of smart phones, an increasing number of people use their mobile phones to read news. In the last decades a lot has changed in the way news articles are presented, but finding interesting news is still a cumbersome activity. This thesis attempts to overcome this problem by exploring new ways of discovering news.

In the context of this thesis a mobile news application is developed which aggregates news from various news providers and recommends news matching the user's interests by analyzing the user's behavioral pattern when reading news. Moreover the news application's social features enable the user to insert comments into the article's text and propose new headlines. A content-based news recommender aggregates and categorizes news using the Probabilistic Latent Semantical Analysis algorithm. After publishing the mobile news application, an evaluation of the recommendation algorithms showed that one of the three algorithms performs well while the other two algorithms need a minor adjustment. The social features were hardly used.

Contents

1 Introduction

In the last couple of decades the possibilities to read news have multiplied. While previous generations mainly read the news in the form of printed newspapers, most people are nowadays, with the rise of the Internet, complementing their quest for the latest news stories by reading news online. The Internet has become a ubiquitous and live source for news that lets people choose the news provider and the news they like. In recent years, as smart phones proliferated, more and more people read the news on their mobile devices. This trend together with the emergence of a fast-growing mobile app market brought about a new kind of news reader apps called news aggregators, which unify the news of different content providers in a single app. Although much has changed in terms of news presentation technology, the way people search for interesting news is still the same. Therefore this thesis explores new ways of discovering, reading and annotating news.

This thesis aims at building a novel mobile news reader app that recommends news based on the user's behavioral pattern when reading news. The news application is built along three principles:

- Independence: The application should be able to provide news from various content providers without the need of having to conclude contracts with each news provider. Thus news providers can be comfortably added and removed. However, one must care of complying with copyright laws, while trying to reach independence. The copyright problem is sidestepped by having the phone download the news article from the news provider's website.
- Easy to use: When developing an application that should be widely used, it is crucial that the user interface is intuitive, clean and simple. There shouldn't be too much buttons either and the functionality of each button should be obvious.
- Social: The application should include features that enable users to express themselves by inserting comments right into the article's text and by proposing new headlines. The comments will be visible for other users and can be rated up or down.

1.1 Discovering news

Mainly people find interesting news by browsing through a collection of news articles. It might be a provocative headline, a neat image or just the topic that catches their attention and induces them to read an article. The task of searching for interesting news is cumbersome and time-consuming. The problem exacerbates when people search for news from several sources that specialize in a particular topic. Another inconvenience is that news from different news providers are presented slightly differently.

The stated problems underpin that there is a need for a news reader application that simplifies the life of its users by aggregating news from various sources and topics, only to recommend the relevant news that the user likes. The news application would behave like a live newspaper that adapts its selection of news articles as the user reads and browses through the news. Although the application should display news that conforms to the detected interests, the recommender should be flexible enough to allow for changes in the user's interests. At the same time the recommender should be capable of recommending popular news, while old stories should be discarded.

1.2 Our contribution

In this thesis a mobile news application that introduces a novel way of discovering and annotating news is developed. In particular the thesis addresses the lack of current news applications to provide the user with news matching his interests based on his behavior with presented news articles. This deficiency is surmounted by implementing a content-based news recommender which considers the user's implicit feedback while reading news. The news articles are categorized by computing the word-document co-occurrence matrix which is subsequently fed to a probabilistic algorithm. This computation is done in an offline fashion and repeated in regular intervals to take into account freshly fetched articles. The news recommendations are computed on the fly using the categorization information calculated earlier. Furthermore the mobile application introduces features that allow users to comment articles in a new way that should encourage users to express themselves.

1.3 Thesis structure

In Chapter 2 the thesis opens with topics related to building a news recommender system. An introduction to news recommender systems is given, followed by techniques to categorize documents. Further, recent news applications are presented. Subsequently, Chapter 3 outlines the main design decisions as well as the challenges. Afterwards Chapter 4 evaluates the recommender system's performance. Chapter 5 summarizes the results and highlights the lessons learned. Finally Chapter 6 gives an outlook on possible enhancements.

2 Related work

Building a news recommender system involves various fields such as machine learning, information retrieval and natural language processing. This chapter begins with an introductory discussion about recommender systems and document classification techniques. At the end of the chapter current mobile news applications are introduced. A focus is on news applications that have a way of recommending news.

2.1 Recommender systems

Recommender systems suggest items that are of interest to a user based on both the user's profile and on either community data or the description of an item. In his work "Hybrid Recommender Systems: Survey and Experiments" [1] Robin Burke distinguishes five different recommendation techniques: collaborative filtering, content-based filtering, demographic filtering, utility-based filtering and knowledge-based filtering. The most popular techniques are collaborative filtering and content-based filtering. Demographic filtering methods assign the user to various groups with different characteristics. Knowledge-based systems suggest items matching the user's needs and preferences. Utility-based systems try to recommend the best available option and are not pursuing the aim of building long-term generalization about the user.

Content-based recommenders rely on the classification of items using features found in them. The user's interests are inferred by considering the features in the items that the user rated. Thus the user can be provided with similar items matching his preferences. News filtering is achieved by using the article's words as features. An example of a content-based recommender can be found in "Using Content-Based Filtering for Recommendation" [2]. A content-based recommender has several drawbacks. The first problem is over-specialization, which limits the user to content similar to what he has already rated. By adding some randomness to the recommendation algorithm this problem can be alleviated. Another problem consists in the gathering of enough information to clearly determine the user's preferences. As a result, if users do not provide feedback on items, their recommendations will yield a poor quality. Also a content-based recommender cannot infer future interests of the user. Benefits of content-based methods encompass their ability to recommend new items or their capability to recommend unpopular items to user with unique tastes [3].

Collaborative filtering techniques build knowledge about items by collecting ratings of items from the user's community to subsequently recommend items to that similar users like. One of the first recommender systems was an experimental mail system, built in 1992, using content-based and collaborative filtering techniques, which record reactions to items from the users' community [4]. The system was developed to filter

out the important mails from a vast number of mails coming from the user's mailing lists. Amazon's "Customers who bought this item also bough this" works with an enhanced collaborative filtering method called "item-to-item collaborative filtering", which was found to scale well while producing high quality recommendations and preserving its fast response times [5]. Another example of a sophisticated and highly scalable collaborative filtering method which was applied to Google News is described in "Google News Personalization: Scalable Online Collaborative Filtering" [6]. While collaborative filtering techniques do not suffer from the problems of content-based recommenders, they exhibit problems of their own. Collaborative recommenders cannot suggest new items until another user rates it or specifies which other items are similar to it. This issue is known as the cold start problem. Also if a user has unseen preferences that do not conform to the aggregated ones from other users, he will not be satisfied with the quality the recommendation. An advantage of collaborative filtering is its suitability for different kinds of items including text in various languages, movies, songs, restaurants.

In an attempt to build a recommender system that exhibits the advantages of content-based recommenders and collaborative filtering techniques, a new breed of recommenders called "Hybrid Recommenders", which combines the two methods, was developed. The study "Hybrid Web Recommender Systems" from Robin Burke evaluates various hybrid techniques and claims that their performance is good [7]. As the recommender's performance significantly depends on the data and application domain, it is not trivial to state which algorithm is the best for a given situation.

2.2 Document classification

In the context of this thesis the focus is on automatic document classification techniques. These techniques can be further split into supervised, semi-supervised and unsupervised methods. "Supervised" means that the algorithm learns how to classify information by processing labeled training data. Thus the wished outcome of the classification process is known in advance which stands in contrast with unsupervised learning methods. As many document classification techniques have been elaborated in the last decades, this section will only cover the basics of a tiny fraction of them all.

Support Vector Machines (SVM) are an example of supervised learning models that separate data points into two sets by finding an optimal hyperplane. This concept can also be extended to patterns that are not linearly separable. A formal, introductory discussion can be found in "A Tutorial on Support Vector Machines for Pattern Recognition" by Christopher J.C. Burges [8]. This work also mentions several limitations of SVM. The speed limitation is only partly solved and requires two training passes.

Furthermore the “training of very large datasets is an unsolved problem”. In addition the author claims that “the optimal design of multiclass SVM classifiers is a further area of research”.

Decision tree learning is a document categorization method that works with classification trees, which enable to infer the category of the document by traversing a tree from the root down to the leaves and at each node following the branch that matches properties, of the document’s content, defined by the node. Although decision tree learning is by definition a supervised classification method, it has been proved to lend itself to unsupervised classification too [9]. Despite the method’s fast data analysis properties even with large data sets, it does not a good candidate for classifying news. A study evaluating the performance of different document classification methods with German texts found that decision tree learning does only qualify for low-dimensional data [10]. In May 1976, Laurent Hyafil and Ronald L. Rivest showed that the construction of an optimal binary decision tree is NP-complete [11] and thus likely of non-polynomial time complexity. ID3 [12] is a notable example of a decision tree generation algorithm.

Naive Bayes classifiers are supervised probabilistic categorization techniques that usually estimate the most probable class by making use of the Bayes’ theorem and by applying the maximum likelihood method. Although the presence of individual class features are assumed to be independent and despite the model’s probabilistic simplicity, the Naive Bayes classifier has proven to be a reliable tool in spam filtering [13], medical treatment optimizations [14] and systems performance management [15]. Naive Bayes is not well suited for text classification according to the paper “Tackling the Poor Assumptions of Naïve Bayes Text Classifiers” [16], which blames the features assumed independence and the neglected consideration of term frequency distributions.

Unsupervised classification methods, which can perform document clustering without any supplementary external information, are presented in section 3.1.1.4 named “News article embedding”.

2.3 News applications

Many news applications exist for the iPhone and for the Android platform. Indeed all major news providers like CNN, Reuters and New York Times have their own news application. These apps provide the same news that can be seen by visiting the editor’s web site. The benefit of using such an app is that the news is presented in a way that makes reading them more enjoyable on a mobile device. In addition to only presenting the news nicer, news aggregators like Pulse [17], Flipboard [18], Zite [19] or News360 [20] are able to combine the news from multiple sources. Pulse is not showing

recommended stories, but only displays recent news from various RSS feeds hand-picked by Pulse's developers. Thus the Pulse app boils down to a fancy-looking news aggregator.

Flipboard, which was like Pulse first released in 2010, mixes news, from social networks and international news, to present them in a magazine-like format. Flipboard's "Cover stories" are chosen by analyzing the user's behavior and recommending stories the user might be interested in. As any documentation is unavailable, it is unclear which algorithms are being taken advantage of to put together the "Cover stories" section. Figure 1 shows Flipboard's home screen.

Figure 1: Flipboard home screen with cover stories

Emerging from research at the University of British Columbia's Laboratory for Computational Intelligence, the Zite news app was built on a solid machine-learning foundation previously used to tag web sites [21]. Zite finds the interests of the users by

sifting content from the user's Twitter or Google Reader accounts. In addition Zite analyzes the user's behavior when dealing with a selection of stories to infer which topics are the user's favorites. The app also utilizes the likes and dislikes users attribute to news articles [22]. CNN has acquired Zite in August 2011.

The new redesigned News360 iPad app, which was released in July 2012 attempts to figure out the interests of the users, by letting them select categories, by analyzing their content on social networks and by evaluating their behavior with presented news [23]. News360 looks at which news the user selects, how long users read news articles and how deep they go on articles [24]. The new app's user interface has been beautified; Figure 2 shows the home screen.

Figure 2: The new News 360 iPad app (Source: iTunes)

3 Architecture

The news recommender system consists of a server that recommends news articles and a mobile client that displays them. The following subsections describe the design process, the decisions that have been taken as well as the challenges and some details about the implementation.

3.1 Server side development

The server's mission is twofold, on the one hand the server must regularly fetch the newest news stories and on the other hand it has to listen to requests for recommendations. The two tasks are achieved by individual applications, in fact the first dubbed the "News aggregator and categorizer" is a standalone Java application and the second called "Recommendation engine" is a set of servlets.

Although the servlets and the Java application have separated control flows, they are using the same database. The servlets are accessing database tables that the Java application is manipulating. In particular when the servlets compute distances in the multi-dimensional Euclidean space there could be scenarios in which requested data is not yet available or going through an update phase. As halting the servlet's operation until the database is fully updated is not an option, another solution needed to be found. Simply initializing the coordinates with the value zero proved to be a good solution since the update of the records just lasts for about 15 seconds and the induced error introduces additional randomness into to the recommendation algorithm. The induced error is small because updates to the article's coordinates happen frequently and therefore the number of newly added articles is only around 10-15 articles, which are not going to have a significant impact on coordinates computed from a collection of 8'000 articles.

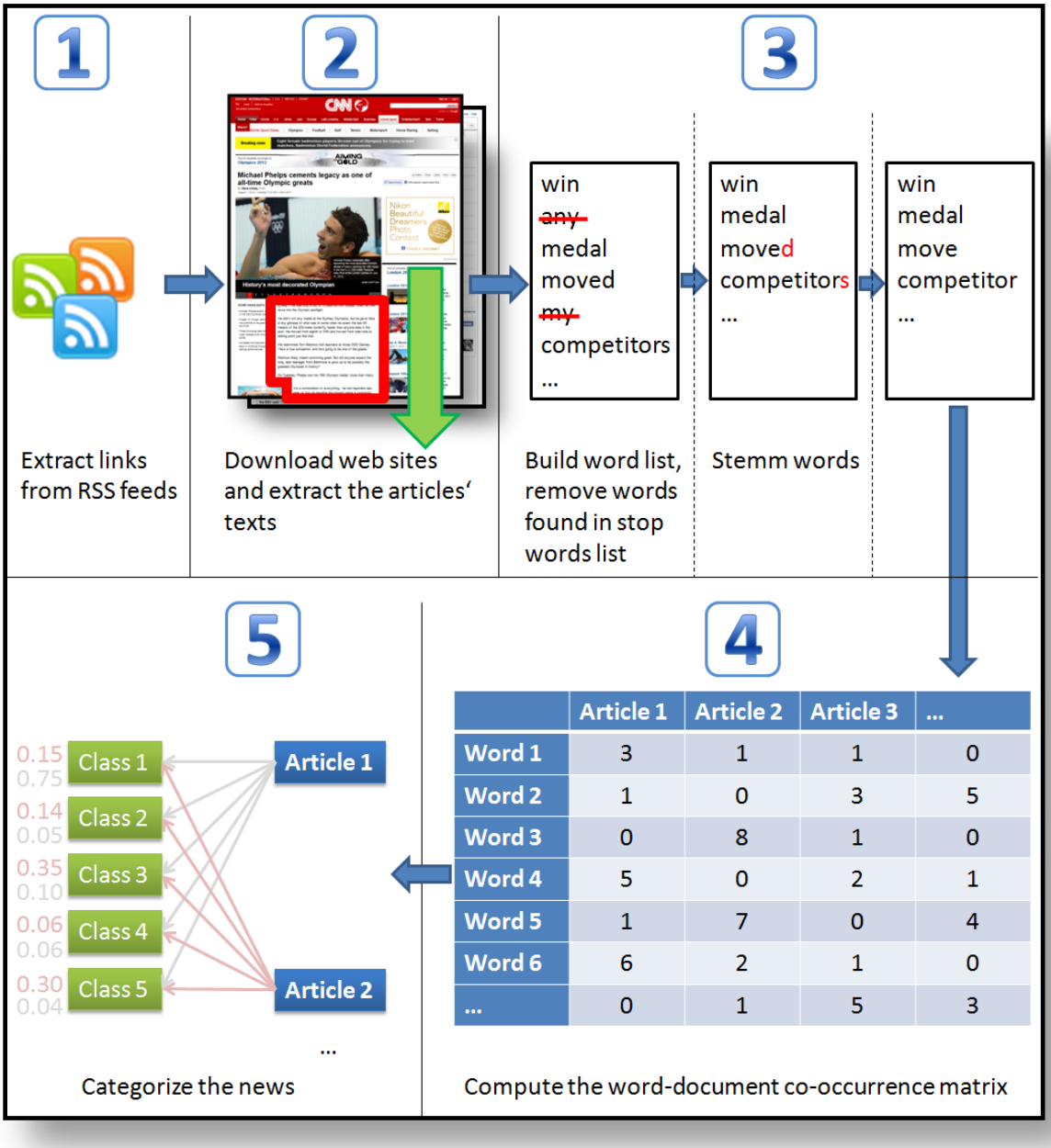
3.1.1 News aggregator and categorizer

The operation of "News aggregator and categorizer" comprises the following steps which are executed sequentially:

1. Extracting links to the latest news articles from RSS documents of various news providers. The list of links to the RSS documents is preconfigured in the program code and can easily be modified or extended.
2. Downloading the news content using the links of the previous step. Followed by the extraction of the actual news content: title, URL of the main article image, article's text, author of the article.
3. Decomposition of the article's text into an array of words which are filtered and stemmed
4. Computation of the word-document co-occurrence matrix

5. Categorization of the news content by feeding the word-document co-occurrence matrix to an implementation of the PLSA algorithm

This operation is scheduled for execution on a regular basis. Figure 3 illustrates the operation of the news aggregator and categorizer. In step five the probabilities of an article belonging to a specific category is listed on the left side of a class. Note that for a specific article, the sum of the probabilities, that this article belongs to a certain class, computed over all classes is equal to one.



	Article 1	Article 2	Article 3	...
Word 1	3	1	1	0
Word 2	1	0	3	5
Word 3	0	8	1	0
Word 4	5	0	2	1
Word 5	1	7	0	4
Word 6	6	2	1	0
...	0	1	5	3

Figure 3: Operation of the news aggregator and categorizer

3.1.1.1 News retrieval

Before starting to fetch any news one must think about where to get them from and which news are being focused on. We decided to focus on international news, because our mobile application will be available globally and we would like to treat all our users equally wherever they are. It is not feasible for us to provide location-specific news to all users, as we would have to include major news providers for every conceivable country. Further, if only a subset of users would receive targeted news specific to their location, they could possibly like the provided news more or less than the other users, thus biasing our evaluation of the recommendation algorithms. As our application will be used globally, it will only provide news in the English language from popular news editors. Among others topics include politics, business, sports, entertainment and technology.

The social media monitoring company Trendiction S.A. gave us access to their service named Talkwalker that allowed us to receive up to date, structured and sanitized news content from the sources we configured. In fact their structured news article content subdivides a news article into title, text, published date, influrank and other metadata. The influrank information is a measure of importance, which considers the relevance, visibility, engagement and quality of an article [25].

Initially the Talkwalker technology was used on the server to aggregate news content, only to be replaced later by our own news retrieval and content extraction technology. By using our own technology, the developed news recommender system is independent and our content extraction technique allows a much more fine-grained and customizable extraction of the article's elements. Moreover the article data from the Talkwalker doesn't reveal any details about the structure of the article, information that is crucial for the content extraction on the phone. The information about the article's structure is defined by the extrVersion number explained further in the next section.

As already mentioned the latest news stories are found by regularly checking a list of RSS documents. If new links are discovered in an RSS document, the news article is downloaded and the content is extracted. This technique could be problematic, as news stories that are updated and do not change their URL will not again be updated on our server. Fortunately updates to a news article will not completely change the meaning of that article and therefore our content-based recommender can still ensure a correct operation.

3.1.1.2 News content extraction

Many techniques have been evaluated to properly handle the task of news content extraction. This section presents attempts to extract news content with several tools and explains the adopted choice. News content extraction aims at taking out the essential parts of a news article from the news article's web page. Among others the essential parts encompass elements such as the title, the article's text and the author's name. Furthermore the content extractor should be fast, should run on Android and should be customizable in order to cut out arbitrary information from a wide range of web sites.

Boilerpipe [26], which was first tested, is a Java library that is able to extract the main textual content from a website using algorithms that are based on concepts of the paper "Boilerplate Detection using Shallow Text Features" [27]. Although the library is fast, simple to use and usually accurate, it doesn't qualify for our needs as customization is cumbersome and the returned text contains all the extracted information in a single block.

Web-Harvest [28] is another more powerful Java library that allows scraping content from websites. The library can be comfortably included in Java projects and is configurable through extraction instructions in XML files. This sounds like the perfect tool for our undertaking, but unfortunately Web-Harvest could not be made to run on Android. The library is also too heavy-weight for the use on a mobile device.

Another option is to build our own scraper. The task of scraping can be divided into several stages, involving the download of the HTML content, a subsequent sanitization and DOM (Document Object Model) parsing process and finally the selective extraction of content from the generated DOM. Many libraries exist to perform the first two subtasks, the most prominent of which are TagSoup and jSoup. jSoup [29] was chosen because it also supports CSS and jQuery-like selectors and runs on Android. This choice narrows down our remaining work to systematically cutting out information of the generated DOM. Such an implementation has been developed by Dominik Bucher in his work "Information and Entertainment Application for Android and iPhone" [30]. His implementation, being platform-independent, runs on both Android and iPhone. The platform-independence is achieved by writing Java code using the Google Web Toolkit (GWT) [31] which translates the code to JavaScript code and the latter is then fed to PhoneGap which generates an iPhone and an Android app. Therefore his code had to be translated from GWT code to standard Java Code. His code heavily relied on the GWT library GWT Query [32], fortunately jSoup was found to be a perfect replacement. GWT

Query, is a jQuery clone, which was primarily used in the GWT implementation for its ease of use to deal with CSS selectors. jSoup is compliant with the exact same selectors.

Moreover some websites like Gizmodo.com build up the main page content by manipulating DOM elements. To extract the main page content, the scraper must render the page after downloading it and before parts of the DOM are extracted. The rendering issue was solved by creating a C++ Qt [33] console application that loads the web page in a Qt WebView to get the rendered DOM from it. The Qt application is called from the Java program, writes the rendered DOM to a file which can in turn be retrieved by the Java program to extract content from it. What further increases the complexity of this solution is the Qt console application's need for a window system, since it encompasses a WebView. So as to run the console application on a headless production server the tool Xvfb [34], which emulates a virtual window system by performing graphical operations in memory, was put to use. This solution was not adopted in the final implementation of the server since there are only few websites that make use of JavaScript to display the main content on their page and secondly the news extraction process would have taken too much time (around 10 - 15 seconds per article).

Our scraper requires XML configuration files that give instructions about which contents should be extracted. However news providers may have a different page structure for every news topic they serve. Therefore the configuration files reside in folders with names specific to their news provider and are named after a number called extrVersion. This naming convention makes it possible to have as many configuration files as there are differently structured news pages for a single content provider and also allows serving configuration files of distinct extrVersion numbers independent of the news provider.

When a new article is added to our server's database the extrVersion, pointing to the needed configuration file to extract the content of the page, is also stored. Each RSS feed of a news provider links to news content of a specific topic, it may be that the page structure varies across those topics. It was observed that most of the time the page structure of news content indexed within the same RSS document was the same. Thus for each RSS document that is followed a configuration file must be provided that is able to extract the news content.

3.1.1.3 Word counting and stemming

As already outlined earlier, the article's text is decomposed into a list of words which is filtered, the remaining words are stemmed and the article is finally categorized in the last step which is discussed in the following section.

The list of words is filtered by excluding words that are found in a stop words list. The stop words list consists of words that do not directly contribute to the meaning of the processed text and can therefore be omitted. The list of stop words includes words like: the, a, or, and, we, necessary, likely.

Stemming is the process of reducing a word to its root form. It is necessary to stem the article's words for the subsequent occurrence counting of each word, as this significantly reduces the number of different words found in the text and also ensures that words having the same meaning are considered as such. For example, the stemmer would transform the words "sings" and "singing" to "sing" or the words "flying", "fly", "flies" to "fli". The Porter Stemmer [35] was selected for the task as it is the most prominent and most widely used English stemmer. Martin Porter, the developer, has also released an extended version of the original stemmer application which is capable of stemming words from various languages including French and German.

The word counter builds up the word-document co-occurrence matrix. The matrix's columns stand for the documents (articles) and the rows for the words in the concerned documents. Each entry in the matrix assumes the number of occurrences of the pertaining document and word. The matrix is generally very sparse and will therefore be represented in memory by only retaining non-zero values.

To speed up the word processing of the article texts, all database operations only take place before and after the processing of the articles. In the beginning, the complete list of words from the words table and the article's texts, that need to be analyzed, are loaded into the memory. Right after the processing of the articles the result is written back into the database. This technique requires the creation of data structures similar to the rows of the articlewords table and the rows of the words table, additionally the data structures must properly increment the wordIDs. This improvement tremendously boosted the performance, bringing down the processing time of 1'650 articles to three minutes from originally 35 minutes. This is mainly due to the wasteful usage of repeated database queries that checked if a word already existed in the word list, with the improved technique the check is done in memory.

3.1.1.4 News article embedding

Embedding news articles in a multi-dimensional space paves the way for subsequent news similarity analyses. As the columns of the word-document co-occurrence matrix can be thought of as high-dimensional vectors for the news, the article embedding algorithm might be able to find a lower-dimensional embedding. Many techniques exist for the task of algorithmic document classification. While Chapter 2 discusses several

classification techniques that were evaluated, this section presents the method that was selected for the implementation together with related considerations.

The document classification method called “Latent Semantic Indexing” which is also known as “Latent Semantic Analysis” (LSA) [36] tries to find out the hidden meaning of co-occurrence data. The main idea behind LSA consists in computing the cosine similarity between documents to assess how related they are. Through singular value decomposition (SVD) LSA finds a low-rank approximation. Classification can be achieved by querying a document for pseudo-document vectors which are representative of a specific category. LSA, which was first described in 1987, has been at the basis of a more elaborated technique called “Probabilistic Latent Semantical Analysis” (PLSA). PLSA, which was proposed by Thomas Hofmann, boasts a solid statistical foundation and defines a proper generative model [37]. Automated document indexing is achieved through PLSA’s usage of a generalized Expectation Maximization algorithm. A key strength of PLSA is it’s suitability for processing large data sets, as PLSA scales linearly with the number of data points and the number of latent classes.

Being published in 1999, PLSA has been applied to various problems. Its power was demonstrated in areas such as document indexing [38], topic-based document segmentation [39], image auto-annotation [40] and image analysis [41]. The stated benefits of PLSA, especially its good scalability, prompted us to use it for the classification of news.

A PLSA implementation in C++, which has been developed by Samuel Welten in his Master Thesis entitled “Personalized Organization of Music on Mobile Devices” [42], was used to classify the news stories. This implementation required the word-document co-occurrence matrix, the number of latent classes and the number of iterations as input arguments. In the ideal case the PLSA algorithm would iterate until the log-likelihood of the PLSA model converges. As an exact convergence might never happen, an appropriate number of iterations must thus be supplied. A simulation of the log-likelihood of the PLSA model in which the number of classes was varied from 0 to 500 (Figure 4), revealed that the log-likelihood function was strictly monotonically increasing and that a number of latent classes greater than 100 was unnecessary. Therefore the number of latent classes was fixed to 100. After the PLSA processing, the news articles can be represented in a multi-dimensional Euclidean space, which allows to measure distances between articles in order to assess their pairwise similarity. The embedding in the Euclidean space is a premise for the recommendation algorithms that are treated in detail in the next section.

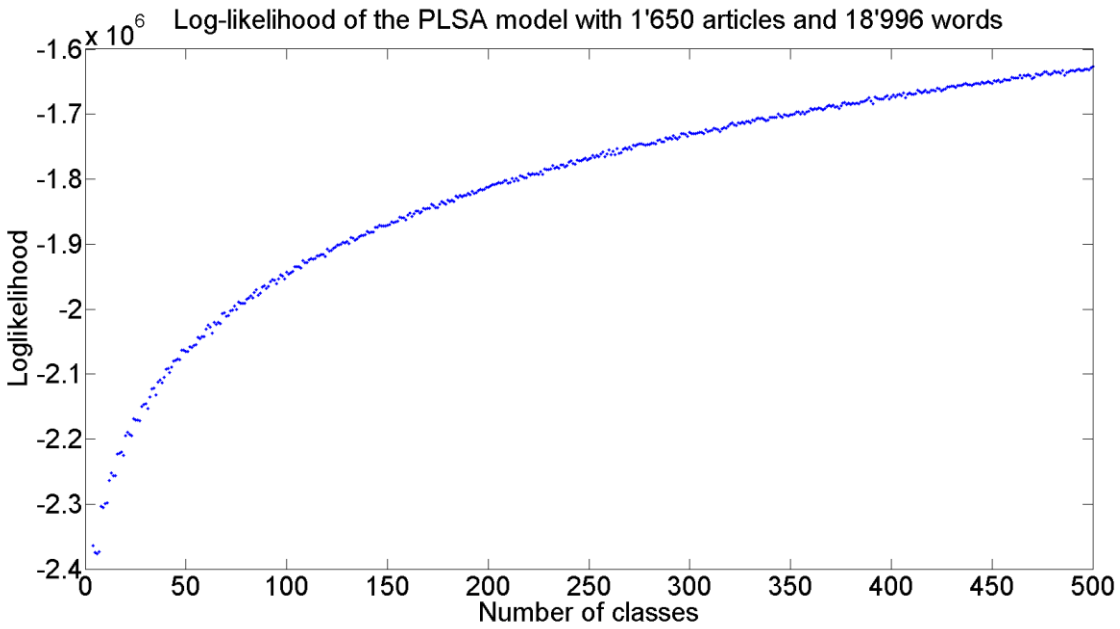


Figure 4: Log-likelihood of the PLSA model as a function of the number of latent classes

3.1.1.5 Database structure

The database is made up of the following 12 tables:

- articles — Stores the articles together with the extrVersion number
- articlecategories — Saves the articles' coordinates in the Euclidean space computed by the PLSA algorithm
- providers — Maps providerID to a provider's name
- articlewords — Contains the table representation of the word-document co-occurrence matrix.
- words — Saves the word list and maps wordIDs to words
- wordcategories — Records the words' coordinates in the Euclidean space
- users — Lists all the users with their usernames and IMEI
- userrating — Contains the ratings of all rated articles
- articlecomments — Saves all the comments with their position in the article's text
- usercommentrating — Has the ratings of the comments
- usertitles — Stores the headlines proposed from the users
- usertitlerating — Records the ratings of the usertitles

3.1.2 Recommendation engine

The recommendation engine computes recommendations based on both the user's preferences of already rated news articles and the news coordinates in the multi-dimensional Euclidean space. Initially, when the user first requests a recommendation, his interests are unknown, forcing the recommender to choose a random article. This insufficiency of information about the user is known as the cold start problem. As the user rates news articles, his preferences become clearer thereby improving the user-dependent perceived quality of the recommended content. The choice of the rating method is another factor that significantly influences the quality of the recommendations. The rating method tries to calculate a measure of inclination for the presented news article, by observing the user's feedback. We found that measuring the amount of time a user spends on a news article was a great way to determine the user's interest in that piece of news. More on the discussion of finding an appropriate rating method can be read in the section "Rating method".

Furthermore, the news recommendation system should filter out duplicates in order to prevent that the user gets the same news story more than once. Filtering the news stories by URL proved to be a sufficient method to handle the issue of duplicates. Additionally one could have excluded news stories that are too closely related to an already rated story.

The recommender's servlet GetRecommendation is capable of delivering multiple recommendations in a single response. This functionality is used on the mobile device to more quickly fetch several recommendations while buffering the news articles. Without any further precautions it might happen that the recommender suggests the same article more than once, the reason being that the articles in the buffer have not yet been rated and as the recommender has no way of knowing that those articles have already been proposed, they might be recommended again. To avoid such an unwanted behavior, the recommender records the recommended articles by saving them in the database with a rating of zero. When the user later rates the article the rating value is overwritten. If the user does not rate the recommended articles, the valid field of rows with a zero rating is automatically set to false after a timeout of a few hours. The persistence of zero ratings could otherwise negatively affect the quality of the recommendations.

A recommendation is an XML document composed of one or more story elements, which contain attributes like the article's ID, the URL to the story, the content providers name, the published date and the extrVersion number that identifies the extraction instruction file suited to extract the news content from the web page. In this way no

news content is distributed from our servers and the mobile device can source the content from the news provider. The story elements from the GetTagRecommendations servlet additionally include the keywords from the recommended articles. More information about the recommendation servlets can be found in Table 1.

3.1.2.1 Recommendation algorithms

The recommendation algorithms should be capable of suggesting news that are matching the interests of the user, are popular among the other users and are not as old as that people don't care about them anymore. Six algorithms are presented next, which basically stem from three different algorithms that can optionally be randomized in the last phase of their three phase algorithm.

After each algorithm description a figure illustrates the algorithm's operation. For reasons of presentation simplicity the articles are presented on a two dimensional grid. Articles could be placed anywhere on the grid, but to make it easy to measure distances the articles occupy positions at the intersection of vertical and horizontal lines. Black dots mark rated articles, whereas blue dots indicate randomly selected articles that have not yet been rated by the user. The article that is selected at the end of the algorithm is shown with a red square around its dot.

Algorithm 1

- Select n articles randomly among all articles that have not yet been rated by the user.
- For each of these articles choose the nearest rated article and assign its rating as a score to the current article. If there is more than one nearest article, the one which has the highest score is chosen.
- Of all the random articles recommend the one which got the highest score.

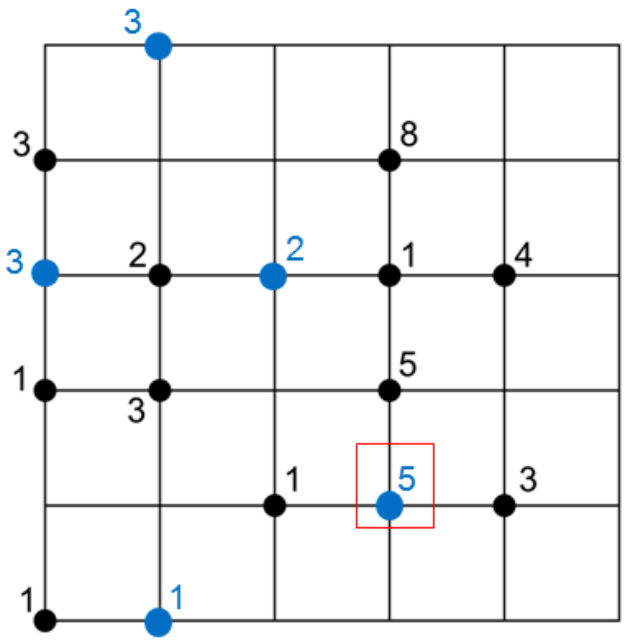


Figure 5: Algorithm 1

Figure 5 depicts "Algorithm 1". Black numbers stand for the rating assigned to the article, while blue numbers denote scores of randomly selected articles.

Algorithm 2

- Select n articles randomly among all articles that have not yet been rated by the user.
- For each of these articles look at all rated articles in a radius of distance R, compute their average rating and assign it as a score to the current article.
- Of all the random articles recommend the one which got the highest score

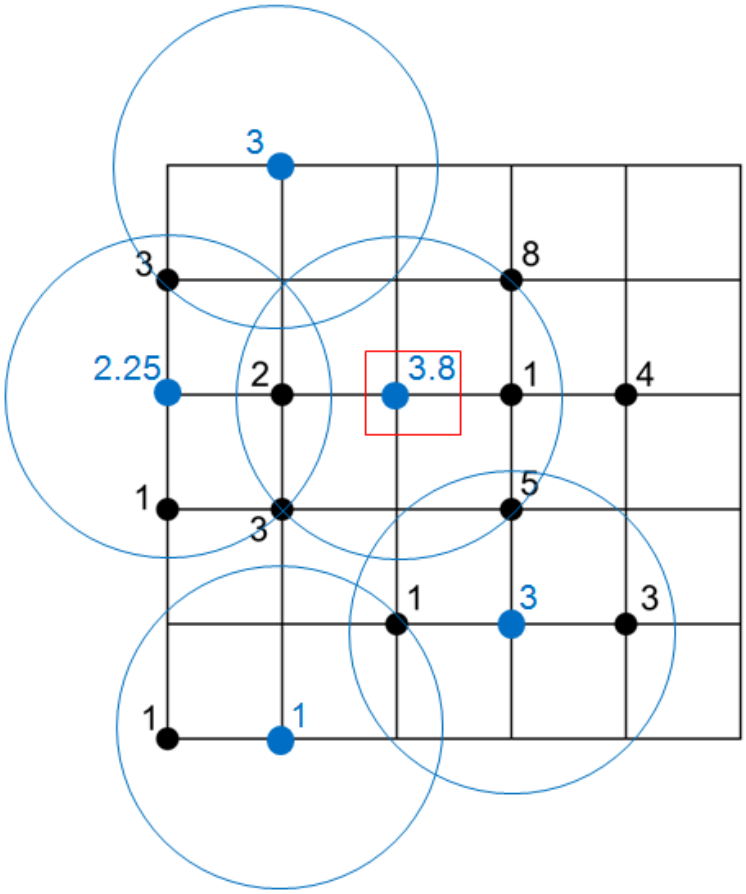


Figure 6: Algorithm 2

The working of "Algorithm 2" is shown on Figure 6. The blue circles delimit the area that is searched for rated articles, which are subsequently used to compute the average rating. The average rating is represented by the blue number next to an article's blue dot.

Algorithm 3

- Select n articles randomly among all articles that have not yet been rated by the user.
- For each of these articles look at all rated articles in a radius of distance R, compute their average, add to this average 1/2 times the average rating among all users of the current article and assign it as a score to the current article
- Of all the random articles recommend the one which received the highest score

The third algorithm is the only one that additionally considers the feedback from all the users thereby supplementing the recommender with a flavor of collaborative filtering. The operation of this algorithm is shown on Figure 7. The numbers, dots and circles have

the same meaning as on Figure 3. Half the average rating of an article among all users is identified by the green scores. The sum of an article's green and blue scores determines the actual score which decides on the recommended article.

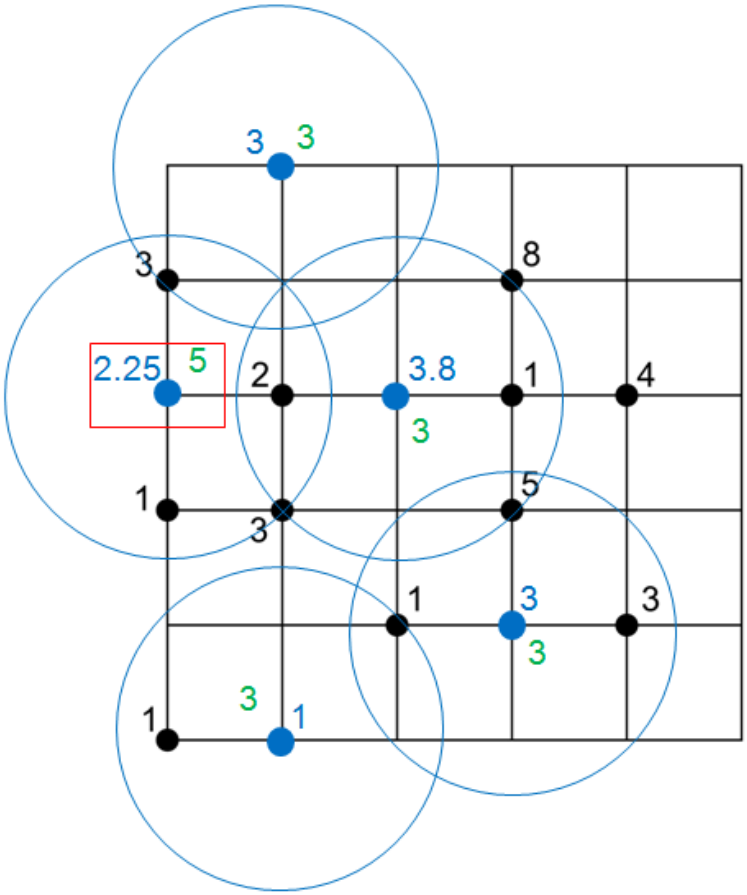


Figure 7: Algorithm 3

As mentioned above, all of the three algorithms can be varied by randomizing the last step, which can then be formulated as follows:

- Recommend randomly, with probabilities proportional to the score, an article out of the random articles

When choosing the random articles, each potential article candidate is selected with the same probability out of the pool of available articles. Every article will only be recommended once for each user, this ensures that users never end-up twice with the same article and once that the user has rated all the articles he will not be given any recommendations anymore.

Users don’t want to be bored with stories of the past. To overcome that problem a fourth phase, which scales the article’s scores according to the article’s age, was sandwiched between phase two and three. The older an article is, the smaller the scaling number gets. A finer article grading is obtained by defining multiple age thresholds. For example scale articles older than one day by 0.5, those older than three days by 0.25 and finally those older than a week by 0.1. This solution still allows older stories to be recommended but diminishes the probability of such an event.

For every recommendation request the recommendation algorithm is randomly selected with each algorithm having the same probability of being selected. The benefit of choosing the algorithm on the server is that we have more control over which algorithms are used. This is not the case for the mobile app, as many users do not regularly update their apps, which makes propagating changes in the mobile app a lengthy undertaking. In the final recommender implementation only the three recommendation algorithms with a deterministic third phase were retained. The number of algorithms was decreased in the hope that it might render the results of the performance evaluation of the algorithms more meaningful in the case of a small amount of rating data from the users.

3.1.2.2 Servlets

The servlets in Table 1 have been implemented to fetch recommendations.

Servlet	Arguments	Description
GetRecommendation	phoneIDNr: the phone's IMEI number (String) articleID: articleID of the rated article or -1 if nothing is to be rated (int) time: the amount of time spend on an article in 1/10 seconds (int) recommend: if a recommendation should be returned (boolean) algo: the algorithm that was used to recommend the article that is to be rated number: the number of recommendations to return	Rates a single article and returns one or more recommendations
GetTagRecommendations	phoneIDNr: the phone's IMEI number (String) articleID: articleID of the rated article or -1 if nothing is to be rated (int) time: the amount of time spend on an article in 1/10 seconds (int) recommend: if a recommendation should be returned (boolean) algo: the algorithm that was used to recommend the article that is to be rated (int) **number**: the number of recommendations to return (int) randomselected: whether the article that is to be rated was a random one (boolean)	Rates a single article and returns **number** recommended articles and **number** random articles together with three keywords for each article. The keywords consist in words from the article that have the highest occurrence count.

Table 1: Recommendation servlets

The social features rely on the servlets from Table 2.

Servlet	Arguments		Description
GetSocialData	phoneIDNr: articleID:	the phone's IMEI number (String) the current article's ID (int)	Returns the comments and proposed titles for a specified article. This servlet is used to refresh the social information of articles stored on the mobile device.
GetUsername	phoneIDNr:	the phone's IMEI number (String)	Returns the user's username associated with his IMEI number
UpdateUsername	phoneIDNr: username:	the phone's IMEI number (String) the new username (String)	Updates or sets the username of a user
PostComment	phoneIDNr: articleID: comment: position:	the phone's IMEI number (String) the current article's ID (int) the user's comment (String) the position of the comment in the article's text (int)	Adds a comment to the database
PostUsertitle	phoneIDNr: articleID: usertitle:	the phone's IMEI number (String) the current article's ID (int) the user's proposed headline (String)	Adds a headline, proposed by the user, to the database.
RateComment	phoneIDNr: commentID: rating:	the phone's IMEI number (String) the comment's ID rating for the user's comments. Up or down, respectively 1 or 0	Rates comments from users
RateUsertitle	phoneIDNr: usertitleID: rating:	the phone's IMEI number (String) the proposed headline's ID rating for the user's headlines. Up or down, respectively 1 or 0 (int)	Rates headlines from users

Table 2: Servlets for the social features

3.2 Client side development

A mobile news reader application for Android devices was developed to visualize the recommended news and detect the user's interests. The application targets mobile phones but also works on Android tablets. Because the news recommender identifies the user's mobile device by its IMEI (International Mobile Equipment Identity) number, the recommendations given to tablets, which may not have an IMEI number, are not accurate. Nevertheless the IMEI number was chosen to identify the user since retrieving the device ID might not reliably work on all devices and is even subject to change in the case of a factory reset.

The application's architecture conforms to the Model View Controller (MVC) design pattern. MVC prescribes a clear separation between the controller, the view and the model of the application. The controller is responsible for capturing the user's actions, inform the view about the actions and update the model's state. The view's operation consists in managing the visual presentation of the user interface. Lastly the model's action is confined to the application's business logic. This separation of concerns renders the components reusable and more flexible.

The mobile application communicates with the server by sending HTTP GET or POST requests and receiving XML documents as a response. All the network communications are running in background threads utilizing Android's AsyncTask class. Android mandates to perform the execution of long-running operations, which could potentially block the user interface, in a background thread. Therefore the database operations, working with the built-in SQLite database, are also executed in a background thread.

3.2.1 User interface

From the very beginning the aim was to build a mobile application with a very intuitive and easy to use user interface. The news articles should be presented with a clear layout and the content would only include the news content like the title, a main article image, the published date, the provider's name and the news article's text. Originally the app was conceived to load and show a single news article at start-up. By dragging the current article to the left, the next recommended news story situated on its right would show up. To get back to read articles it would suffice to drag the current article to the right. Such a natural way of browsing through the news should permit the users to discover news without having to take any decisions. Figure 8 shows the home screen and the "reading view" of the mobile application.

As the mobile application evolved through the user's feedback, more features, like the capability to browse news by tags, were added. Also a main menu which is shown on start-up was created. The functionality that was initially hidden in the context menu was

moved to the main menu. Through this menu all important features are seen at a glance which renders the application more easy to use and makes users aware of all the functionalities. Further, a top of the screen menu bar with the application's logo was created to ensure that the application has a unified design throughout the different screens. Moreover the font size was made adjustable as some users were not able to read the news articles easily.

Text justification enhances the look of the news article's text. As full justification is not supported by Android's TextView widget, a WebView with CSS formatted text was employed instead. The WebViews's scrolling and zooming capabilities had to be deactivated to give it the same look and feel as a TextView.

Figure 8: The NewZap home screen and the NewZap reading view

3.2.2 Rating method

As mentioned above, choosing a good rating method is of outmost importance for the recommender to work efficiently. There are several possibilities to figure out how much the user likes a specific piece of news. One could present a five star rating bar to the

user. Some users might argue that five stars are not enough, others might complain that they cannot decide how many stars to rate the article. As a result most people would not rate any articles. The people could be overwhelmed by the number of stars they see, so just give them two choices: “Thumbs up” or “Thumbs down”. But even with just two possibilities, articles wouldn’t be rated at all times. The amount of information from such responses would also not be precise enough to infer the user’s interests. Measuring the time a user spends on an article does away with the described problems, as this metric is always automatically recorded, provides fine-grained rating data and is objective. Thus the time a user spends on an article was chosen as rating function.

The independence of the rating function, with respect to the length of the article, is justified by the consideration that the user may not spend much more time on an interesting, lengthy article than he might spend on a shorter one. Also if the article is quite short, it might happen that the computed rating reveals an inclination for the topic although the user is not interested in the article and was simply too slow to switch to the next article.

Images should also be considered in a way, as they could constitute a source of distraction. Imagine a user pauses on a news article for a long time just because of a beautiful picture. The picture should be representative of the article’s text. According to a study which also analyzed news, the ideational meanings in texts and images relate [43]. Therefore if the user is interested in a certain picture, the recommender’s analysis of the text next to the picture will lead to recommendations with similar texts, which will as a result contain relevant pictures.

As seconds are not fine enough for the rating, the time is measured in 1/10 seconds. Further, there should be an upper limit for the rating. Otherwise a user, that left his mobile device unattended for a longer time and later reads the next news story, will develop a huge interest in the forsaken article.

3.2.3 News content extraction

The news content extraction on the mobile device works in a similar fashion than on the server. Thanks to jSoup’s compatibility with the Android platform, the scraper code from the server could seamlessly be integrated into the mobile application without making any changes to the code. Using this scraper, the various news elements such as the title, the text and the URL to the main image can be retrieved. As the mobile version of the news article might be split among multiple pages for improving the readability of a long article on a small screen, the desktop version, which displays the whole news content on a single page, is fetched instead. The scraper could also have been configured to download the mobile version of the news article and follow the links to

fetch the next part of the article, but obviously this solution would have been slower and would have complicated the extraction instructions. The Android developer documentation does not indicate which platform version uses which user agent string, but sniffing HTTP connections revealed that Android 2.3.3 sends requests with a user agent string containing the word “mobile” whereas Android 4.0 has a user agent string that does not contain the word “mobile”. According to an article [44] published on computer world, which is an IT magazine, the word “mobile” was removed from the user agent string starting from Android 3.0, named Honeycomb. Google states that web servers discovering the word “mobile” in the user agent string should serve the mobile version of their web page [45]. Thus, to ensure that the mobile application works flawlessly on all platform versions, the user agent string has to be set accordingly.

However this solution has limitations when it comes to extracting content from web pages that render the article’s text with JavaScript. A simple download of the web page will not lead to the execution of the JavaScript code to build the modified DOM and hence the news content cannot be extracted. Therefore the web page is loaded into an Android WebView from which the rendered DOM can be obtained. To retrieve the rendered HTML content a JavaScript interface has to be attached to the WebView and secondly a WebView client, which will inject JavaScript code when the page has finished loading, has to be configured for the WebView. The injected JavaScript code has to be programmed such that it grabs the whole HTML code of the page and provides it to the attached JavaScript interface, where the HTML code can be further processed. As this operation takes place in a background thread, a handler should be registered beforehand to return back to the main thread in case the view should be manipulated. Extracting HTML from a WebView is a relatively sophisticated undertaking that only succeeds because of a series of tricks. Furthermore the WebView was observed load pages slowly and at an unpredictable pace. Even, when turning on hardware acceleration and setting the rendering priority to high, the WebView’s slowness persisted. All the mentioned deficiencies of the WebView resulted in the rejection of the described solution and the adoption of a solution which downloads the page’s content without rendering its JavaScript. Since most news websites to not make use of JavaScript to render their news article’s text, such a solution is tolerable.

3.2.4 Swipe through the news

The initial idea of the mobile application envisioned a simple user interface that should provide a way for the user to discover news by swiping through them. Such a user interface is reminiscent of traditional paper-based newspapers, which are read by flipping the pages. The desired user interface consists of multiple adjacent views that can be dragged sideways, presenting only one view at once. This functionality was first

obtained by having three Android ViewFlipper views that were shifted by swiping left or right. A drawback was the lagging responsiveness to user gestures, the view would only move once the gesture was completed. Moreover an animation which defines the disappearance of one view and the emergence of another has to be supplied. In addition to that every time the view changes, the surrounding views have to be updated. Fortunately Android provides another newer view, called ViewPager, which tremendously simplifies showing adjacent views that can be swiped. The ViewPager's inherent capability to display adjacent views that can be swiped and its efficiency in managing a collection of views in memory make it a perfect choice. When swiping, the ViewPager's view follows the finger instantly.

As people should have the possibility to quickly move from one news article to the next, it is not sufficient that news articles are loaded sequentially. Therefore the application has been developed to ensure that there are always three additional news stories in the buffer. A good way to fill the buffer consisted in fetching four news stories when the swipe activity is first started, later on only a single story is fetched each time the user moves to the next article. Previous tries to fill the buffer involved calculating and retrieving the number of remaining stories each time the user switched to the next article. The number of stories required to completely fill the buffer might be greater than one, provided that the number is computed while some stories are still loaded in the background. Thus a user who quickly swipes through the news will receive more news stories than were originally thought to be buffered. It might even happen that the user buffers more and more stories as he browses the news, thus overloading the phone.

Every time the user is shown a new article the current system time is recorded. Upon switching to the next article the time difference between the current time and the previously recorded time is computed and sent as a rating to the server together with the rated article's ID. As contacting the server twice in a row, once to rate the article and one more time to get a recommendation, is wasteful, the functionality of rating and fetching news was consolidated in a single invocation.

In order to speed up the presentation of a news article even more, usage was made of the Android's internal SQLite database. All shown news articles are automatically stored in the internal database. Later when the user starts up the swipe activity again the last seen story will be deserialized and displayed. This leads to a swipe activity that, if not started for the first time, will immediately show a news article. In the background the activity fetches the next recommendations. The database also stores the comments and the proposed headlines from the users. If a news article is retrieved from the phone's

database, the social content is also queried and put into the social view of the article. Since displaying stale social content from the phone's database is not useful for a social application, the swipe activity always displays the social data and then consults the server for an updated version of the social data, which will replace the shown data. Moreover the article's images are not stored on the phone, because downloading them is fast. Only the URL to the image is saved in the phone's database.

3.2.5 Read news by tags

After releasing a first version of the mobile application which only contained the functionality to swipe through the news, some users complained they would like to have more control over which news they are getting. As a response to the users' wish, they were given a way to select news by clicking on keywords in a tag cloud. The cloud shows 18 tags, which belong to three random articles and three recommended articles. Each article is represented by three words that are grouped together and that have the highest occurrence count in this article. Before the groups of keywords are presented, they are randomly shuffled to prevent that random and recommended articles always occupy the same positions on the screen. By displaying random and recommended articles at the same time, the user's preference for either the random or the recommended articles can be assessed. Sometimes the user may not find an interesting keyword in the tag cloud, for this reason a refresh button was added to the menu bar. The news by tags screen can be seen on Figure 9.

A tag cloud can only live up to its name if the group of tags contains tags with different font sizes. Therefore the reproduced tags can assume three different font sizes, which are chosen according to the tag's degree of affinity for a specific category. Whenever a tag cloud is created, a category is selected at random which gives tags with a stronger affinity to it a bigger font size.

Obtaining the keywords of a specific article to display in the tag cloud is a two step process. First the database is queried for the three words that occur most often in the article. These words are actually stemmed words which have to be transformed into words that make sense. Thus in a second step, the stemmed words are used to find matching words in the article's text. The first word that has the stemmed word as a substring is returned. The article has to be correctly decomposed into its words, to avoid that there are special characters in the resulting words that are shown in the tag cloud. Further it may happen that one of the stemmed words is a substring of another. This could lead to a situation in which the same word is returned twice. To prevent that a word is repeated, the algorithm that looks for proper words using the stemmed words, will search for a later match.

The time a user spends on an article is obtained by measuring the time that elapses from the point in time a selected article appears until the moment when the user hits the back button. The device's back button callback method had to be overridden to ensure that all attempts to exit the application are captured. As soon as the time is calculated, the last article is rated and the tag cloud is refreshed.

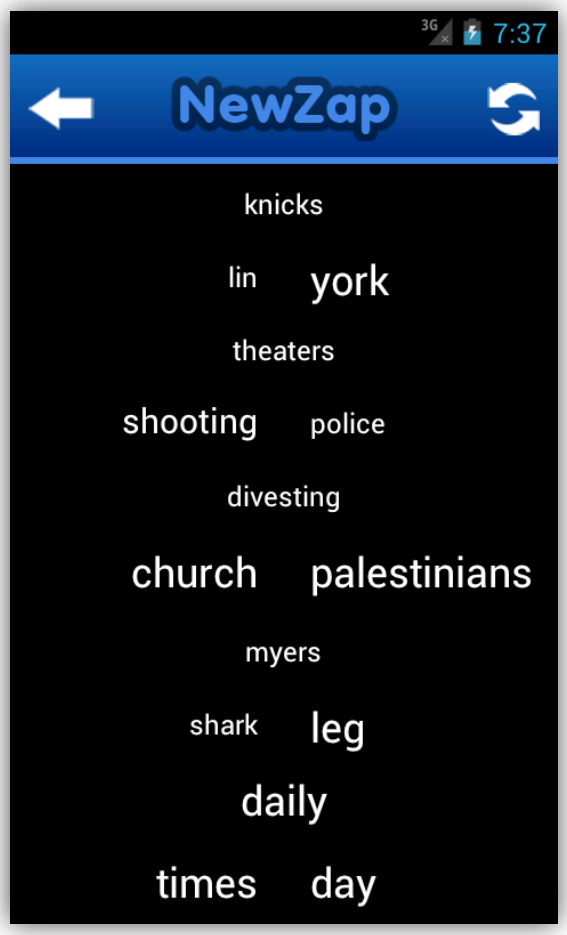


Figure 9: News by tags

3.2.6 Social features

The social features aim to render the mobile application interactive, engaging and fun. Most news websites enable the user to comment on their articles. The comments will usually appear in an area of the page, which is situated below the news article and was reserved for that purpose. When many users have commented before, the user might find his statement at the remote end of a long list of unrelated comments, thereby significantly reducing the visibility of the comment. To improve the visibility, CNN allows sorting the comments by popularity, age and best rating. But sometimes users do not want their comments to show up in a list next to a news article. They would like to insert

their comments into the article, to correct a sentence in the text or to express their opinion about a specific phrase. Moreover the headlines of news articles are sometimes misleading or do not satisfy the reader. An upset reader would like to replace the headline with his own formulation.

The stated shortcomings of current news platforms prompted us to develop a mobile application that allows the user to express his thoughts by putting his comments on top of the article's words right into the article's text. Additionally the application allows the user to propose new headlines that the user thinks are more appropriate for the article. Users can vote the comments and headlines up or down. Of course as to prevent misuse, they cannot rate their own content or vote twice for the same content. A rating method with "thumbs up" and "thumbs down" was chosen because of its simplicity for the user. The article's words that were subjected to comments are highlighted with a bold font and colors ranging from red to green depending on the best rated comment on that word. All the comments and proposed headlines assume a rating of zero when they are created. The color of an article's word turns greener as the rating of the best rated comment on that word is increased. Or the color turns pinker as the best rated comment adopts a rating with a decreased score. Every time a user rates a headline or a comment, the rating's score is increased or decreased by one.

As the user should have the possibility to read news lucidly and optionally to switch on the presence of comments, it was opted for a user interface with two separate views: one view for displaying only the news article called the "reading view" and a second view that shows the article together with the users' comments named the "social view". The two views are back to back, a single click on the social view button in the menu bar flips the view with a short animation. While the reading view is for reading only, the social view has in addition all the features to comment and rate a news article. The social view can be seen on the left picture of Figure 10.

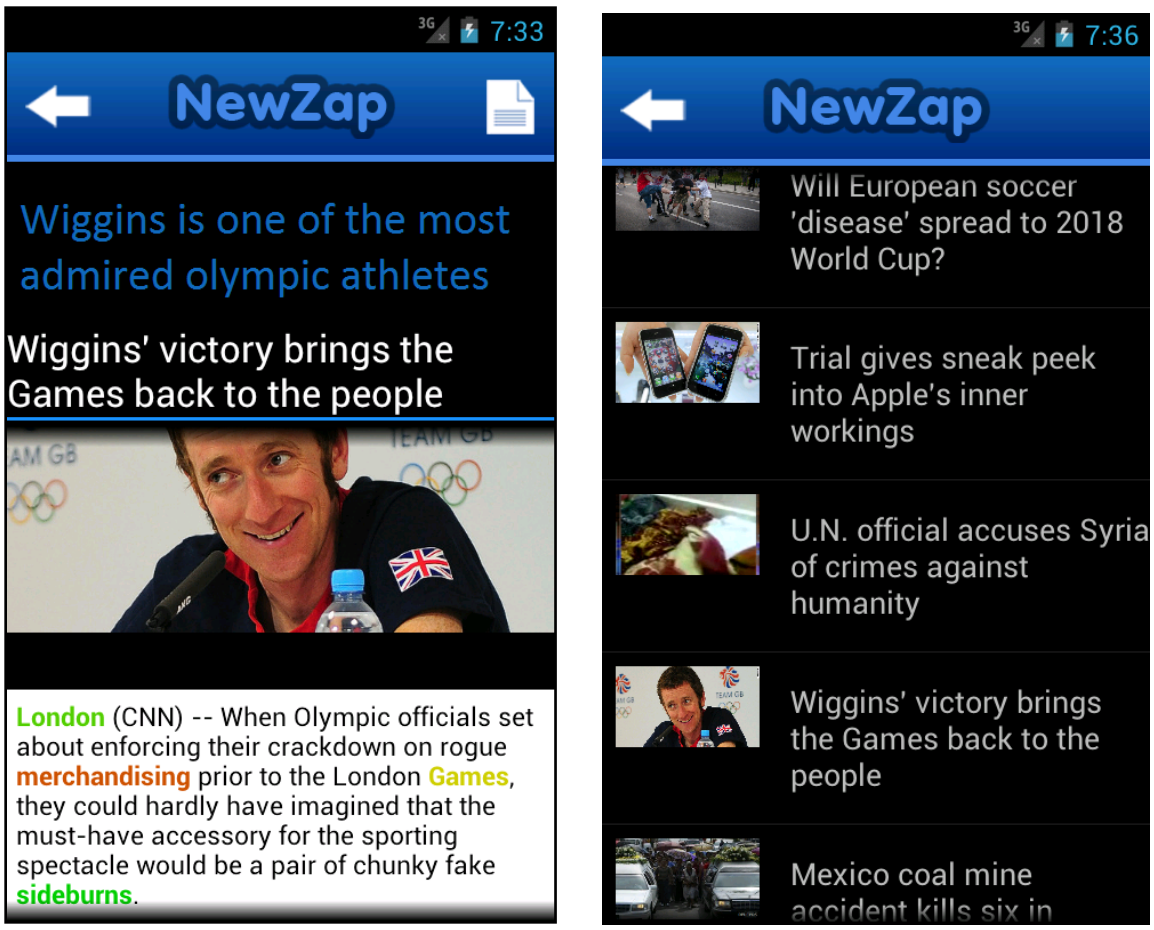


Figure 10: Social view and history screen

3.2.7 News history

The news history enables the user to review the articles that he has already seen. This feature is invaluable since users may want to see an article more than once. Especially if a user commented on an article, he may want to see the feedback from other users or he may want to add another comment. The history shows a comprehensive list of all the news articles that are found on the phone's database. When browsing through the news, the news content is automatically saved in the phone's database. The user's history could also have been retrieved from the server but that would have caused an additional delay, as the stories would again have to be downloaded and extracted before they are shown. Also the creation of a list with headlines of seen articles would have taken a lot of time. Indeed so much time that this solution is unacceptable. Extracting news from a single article can take up to three seconds, thus to show a list with 10 items it would last half a minute to complete the retrieval, not including the 3 seconds that elapse to fetch the URLs of all the stories in the history.

The comments and proposed headlines are also retrieved from the phone's database when article of the history view is selected. To ensure that the user always gets the newest social content, this information is updated in the background and displayed when retrieved from the server. The history screen is shown on the right picture on Figure 10.

3.2.8 Username

A social news application must have a way for the user to choose his username, such that the chosen name can label the comments and proposed headlines of this user. From the main menu the user can change his username at any time and as often as he wants. If no name is chosen the posts from the user will appear with the username "Anonymous".

The username is saved on the server and on the phone. Saving it on the phone brings about that comments and proposed headlines from the user can be immediately shown without waiting for the response from the server, which would contain the username. Nevertheless the username is not available anymore on the phone after the application has been updated. And for this reason the application must request the username from the server whenever the username cannot be found in the phone's storage.

4 Evaluation

The following evaluation aims to shed light on the performance of the recommender system and summarizes how the users have been using the mobile application. The usefulness of the recommended articles is assessed by comparing the average times users spent on articles that were randomly chosen with the average times users spend on articles that were recommended with one of the tree recommendation algorithms.

As already outlined in Chapter 3, the algorithm used to recommend articles is chosen randomly on the server for each recommendation request. Four algorithms to recommend an article exist, either one of the three recommendation algorithms is selected or a random article is chosen. Each algorithm has the same probability of being elected; this probability is equal to 0.25 because there are four outcomes. Initially for the first five recommendations, when not much is known about the user's preferences, a random article is chosen. In every recommendation response the server includes the recommendation algorithm that generated it. When the mobile application rates an article it will also transmit the recommendation algorithm that suggested the article. In this way the server can store the rating together with the algorithm, which enables performance comparisons of the algorithms. Table 3 lists the results of the mobile application's "Zap news" feature and Table 4 presents the results of the "News by tags" feature.

Zap news			
Recommendation algorithm	Average time in s	Number of users	Number of ratings
Algorithm 1	45.15	19	246
Algorithm 2	10.91	17	150
Algorithm 3	11.76	17	164
Random	18.76	26	209
			Total: 769

Table 3: Results zap through the news

News by tags			
Recommendation algorithm	Average time in s	Number of users	Number of ratings
Algorithm 1	9.00	4	16
Algorithm 2	4.71	4	16
Algorithm 3	25.62	2	7
Random	6.84	2	43
			Total: 82

Table 4: Results news by tags

Since every user gets random recommendations in the beginning, the number of users that received random recommendations is the highest in Table 3. When comparing this number with the one from Table 4, one can infer that most users first swiped through the news and only later browsed the news by tags.

From Table 3 one can infer that the recommendations that have been computed with “Algorithm 1” were more interesting to the user than randomly chosen articles. Moreover “Algorithm 2” and “Algorithm 3” performed slightly worse than the random recommendations. Being tightly related, the two latter algorithms performed equally bad. This does not necessarily mean that these algorithms are of minor quality but rather that the radius R was not chosen appropriately. Only a single radius R has been tried out during the measurements. To definitely asses the performance of “Algorithm 2” and “Algorithm 3” the radius R should be varied. Of course, such an evaluation requires a lot more time as enough data has to be gathered for every chosen R.

Table 4 suggests that “Algorithm 3” performs well but due to the small number of ratings this average may contain a substantial amount of noise that may have corrupted the figure. Nevertheless the numbers for “Algorithm 1” and “Algorithm 2” express a similar inclination than their counterparts in Table 3. Moreover by adding up the number of ratings of the three recommendation algorithms in Table 4 (their sum is 39) and comparing them with the number of ratings of random articles (which is 43), one sees that the users clicked with nearly equal probabilities on random and on recommended tags. As the tag cloud contains as many tags from random articles as it contains tags from recommended articles, one can conclude that the users did not favor random or recommended articles by just seeing keywords from them.

Further, one clearly sees, by looking at the number of total ratings in each table, that users preferred swiping through the news to browsing news by tags. The social features have not been widely used. So far 118 users have used the mobile application, of these only 13 have chosen a username. Only five comments have been posted and two headlines have been proposed.

5 Conclusions

This thesis presents the development of a mobile news application that was build along three principles: independence, social and easy to use. The mobile application, which was developed for Android devices, allows users to either discover news by swiping through them or by clicking on keywords in a tag cloud. Furthermore a novel way to annotate news articles was built into the news application.

The developed content-based news recommender system is capable of aggregating and categorizing news from any RSS feed, given that extraction instructions for the news provider are present. Thus new news sources can be easily incorporated into our system. The Probabilistic Latent Semantical Analysis algorithm was utilized to classify the news and to embed them in a multi-dimensional Euclidean space. Together with the amount of time the user spent on different articles, the interests of the user are determined and matching articles are recommended to the user.

The performance evaluation proved that the quality of the recommendations, computed with “Algorithm 1”, is good compared to randomly chosen articles. “Algorithm 2” and “Algorithm 3” have shown a poor performance which is probably linked to a bad choice of the radius parameter. To fully determine the performance of the latter two algorithms the radius parameter should be varied. Ultimately with an adequate choice of the radius, those algorithms should be able to compete with “Algorithm 1”. Therefore simply measuring the amount of time a user spends on an article proves to be a good indicator for the user’s interest in an article.

Although the mobile application’s design was improved multiple times and the users were shown a message about how to insert comments and propose headlines, the annotation features were hardly used. Moreover the users preferred swiping through the news to browsing through the news by tags.

6 Future work

Although the presented news recommender system has been improved until reaching the maturity required to run in a production environment, there is still room for improving the user experience. Some limitations of the current recommender system, which may be addressed in future works, are described next.

Due to the fact that the smart phone market is largely dominated by iOS and Android, an iPhone version of the news application should be developed. Simply porting the Android code to iOS is not possible, since the article content extractor uses the jSoup library, which is not working on the iPhone. A new solution, which is both fast and integrates into the existing framework, has to be found.

In addition, the mobile application could be supplemented with location-specific news. This can either be achieved by letting people choose locations from which they want to get news or by tracking their position using their mobile device's built-in GPS module. With the current recommender system, such an extension would involve changing the news aggregator to assign a location to the news and slightly adapting the recommender to consider the user's location preferences. Figuring out the location of a happening referred to in a news article and deciding whether the news is of international or regional nature are the main challenges.

Now that the recommender system is capable of analyzing and recommending news from any source given that appropriate extraction instructions are provided, the mobile application could be enhanced to allow users to add new content providers themselves. A simple user interface that lets users select the story's elements and generates an extraction instructions file must be developed. Optionally users should have the opportunity to write the extraction instructions files on their own and submit them over a web interface.

The current news recommender system could be used to assess the performance of different recommendation algorithms. Also one could try to determine which radius R would yield good results for "Algorithm 2" and "Algorithm 3".

In recent years the development of personalized mobile news reader applications has gained steam as their popularity is boosted by the proliferation of smart phones and by the people's desire to quickly get the latest, interesting news. This trend will continue as the developers aspire to provide an even richer user experience which makes use of all the unique capabilities offered by smart phones.

7 Bibliography

[1] R. Burke, «Hybrid Recommender Systems: Survey and Experiments,» California State University, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.8200&rep=rep1&type=pdf.

[2] R. v. Meteren und M. v. Someren, «Using Content-Based Filtering For Recommendation,» University of Amsterdam, 2000. [Online]. Available: http://www.ics.forth.gr/~potamias/mlnia/paper_6.pdf.

[3] R. J. Mooney und L. Roy, «Content-Based Book Recommending Using Learning for Text Categorization,» University of Texas, 7 February 1999. [Online]. Available: http://arxiv.org/pdf/cs/9902011.pdf.

[4] D. Goldberg, D. Nicholas, B. M. Oki und D. Terry, «Using collaborative filerting to weave an information Tapestry,» Association for Computing Machinery, December 1992. [Online]. Available: http://www.ischool.utexas.edu/~i385d/readings/Goldberg_UsingCollaborative_92.pdf.

[5] G. Linden, B. Smith und J. York, «Amazon.com Recommendations - Item-to-Item Collaborative Filtering,» February 2003. [Online]. Available: http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf.

[6] A. Das, M. Datar, S. Rajaram und A. Garg, «Google News Personalization: Scalable Online Collaborative Filtering,» Google, 12 May 2007. [Online]. Available: citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.4329&rep=rep1&type=pdf.

[7] R. Burke, «Hybrid Web Recommender Systems,» DePaul University Chicago, 2007. [Online]. Available: http://www.dcs.warwick.ac.uk/~acristea/courses/CS411/2010/Book%20-%20The%20Adaptive%20Web/HybridWebRecommenderSystems.pdf.

[8] C. J. Burges, «Microsoft,» Microsoft Research, 1998. [Online]. Available: http://www.cs.brown.edu/courses/archive/2006-2007/cs195-5/extras/Burges98.pdf.

[9] B. Liu, Y. Xia und P. S. Yu, «Clustering via Decision Tree Construction,» [Online]. Available: http://www.cs.ucla.edu/~wwc/course/cs245a/CLTrees.pdf.

[10] C. Goller, J. Löning, T. Will und W. Wolff, «Automatic Document Classification: A thorough Evaluation of various Methods,» [Online]. Available: citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.966&rep=rep1&type=pdf.

[11] L. Hyafil und R. L. Rivest, «Constructing Optimal Binary Decision Trees IS NP-Complete,» May 1976. [Online]. Available: http://barbra-coco.dyndns.org/eiyou/data/NPComplete.pdf.

[12] J. R. Quinlan, «Induction of Decision Trees,» Springerlink, [Online]. Available: http://www.springerlink.com/content/ku63wm5513224245/fulltext.pdf.

[13] I. Androutsopoulos, J. Koutsias, K. V. Chandrios und C. D. Spyropoulos, «An Experimental Comparison of Native Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages,» National Centre for Scientific Research "Demokritos", 2000. [Online]. Available: http://arxiv.org/ftp/cs/papers/0008/0008019.pdf.

[14] M. J. und K. J., «Application of the Naïve Bayesian Classifier to optimize treatment decisions,» *Radiotherapy and Oncology,* 2007.

[15] J. L. Hellerstein, T. Jayram und I. Rish, «Recognizing End-User Transactions in Performance Management,» American Association for Artificial Intelligence, 2000. [Online]. Available: http://aaaipress.org/Papers/AAAI/2000/AAAI00-091.pdf.

[16] J. D. M. Rennie, L. Shih, J. Teevan und D. R. Karger, «Tackling the Poor Assumptions of Naive Bayes Text Classification,» MIT - Artificial Intelligence Laboratory, 2003. [Online]. Available: http://people.csail.mit.edu/jrennie/papers/icml03-nb.pdf.

[17] A. Gupta und A. Kothari, «Pulse.me,» Pulse, May 2010. [Online]. Available: http://www.pulse.me/products/#/android.

[18] «Flipboard.com,» Flipboard, Inc., 2010. [Online]. Available: http://flipboard.com/.

[19] «zite,» zite, [Online]. Available: http://zite.com/.

[20] «news360,» news360, [Online]. Available: http://news360app.com.

[21] T. Geron, «Wall Street Journal - Venture Capital Dispatch,» Wall Street Journal, 9 March 2011. [Online]. Available: http://blogs.wsj.com/venturecapital/2011/03/09/zite-goes-after-flipboard-with-tablet-digital-magazine.

[22] K. Boehret, «Wall Street Journal - The Digital Solution,» Wall Street Journal, 9 March 2011. [Online]. Available: http://online.wsj.com/article/SB10001424052748704758904576188621188047068.html.

[23] S. Dredge, «The Guardian - News 360 iPad,» The Guardian, 16 July 2012. [Online]. Available: http://www.guardian.co.uk/technology/appsblog/2012/jul/16/news360-ipad-relaunch-news-aggregation.

[24] A. Ha, «Techcrunch - News 360 iPad,» Techcrunch, 10 July 2012. [Online]. Available: http://techcrunch.com/2012/07/10/news360-ipad/.

[25] Trendiction, «trendiction.com - Influrank,» Trendiction S.A., [Online]. Available: http://www.trendiction.com/en/products/influrank.

[26] C. Kohlschütter, «Google Code - Boilerpipe,» 6 June 2011. [Online]. Available: http://code.google.com/p/boilerpipe/.

[27] C. Kohlschütter, «Boilerplate Detection using Shallow Text Features,» in s *WSDM*, New York City, 2010.

[28] «Sourceforge,» Web-Harvest, 17 February 2010. [Online]. Available: http://web-harvest.sourceforge.net/.

[29] «jSoup - Download,» jSoup, 28 May 2012. [Online]. Available: http://jsoup.org/download.

[30] D. Bucher, «Information and Entertainment Application for Android and iPhone,» ETH Zurich, Distributed Computing Group, [Online]. Available: ftp://ftp.tik.ee.ethz.ch/pub/students/2011-HS/BA-2011-12.pdf.

[31] «Google Web Toolkit,» Google, [Online]. Available: https://developers.google.com/web-toolkit/.

[32] «GWT Query,» Google, [Online]. Available: http://code.google.com/p/google-web-toolkit/.

[33] «Qt - Cross-platform application and UI framework,» Nokia, [Online]. Available: http://qt.nokia.com/.

[34] «Wikipedia - Xvfb,» Xvfb - X virtual frame buffer, [Online]. Available: http://en.wikipedia.org/wiki/Xvfb.

[35] M. Porter, «Porter Stemmer Algorithm,» [Online]. Available: http://tartarus.org/martin/PorterStemmer/.

[36] S. Deerwester, «Indexing by Latent Semantic Analysis,» Journal of the American Society for Information Science, 26 August 1987. [Online]. Available: http://www.cob.unt.edu/itds/faculty/evangelopoulos/dsci5910/LSA_Deerwester1990.pdf.

[37] T. Hofmann, «Probabilistic Semantic Analysis,» 1999. [Online]. Available: https://www.mpi-sb.mpg.de/departments/d5/teaching/ss00/proseminar-papers/4/hofmann-sigir99.pdf.

[38] T. Hofmann, «Unsupervised Learning by Probabilistic Latent Semantic Analysis,» 31 July

2000. [Online]. Available: http://webdocs.cs.ualberta.ca/~swang/cs886/hofmann.pdf.

[39] F. C. T. Thorsten Brands, «Topic-based document segmentation with probabilistic latent semantic analysis,» CIKM'02, 4 November 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.4450&rep=rep1&type=pdf.

[40] F. Monay und D. Gatica-Perez, «PLSA-based Image Auto-Annotation: Constraining the Latent Space,» MM'04, 10 October 2004. [Online]. Available: http://infoscience.epfl.ch/record/83106/files/monay-acm-1568937089.pdf?version=1.

[41] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman und W. T. Freeman, «Discovering Objects and their Location in Images,» Carnegie Mellon University, 1 January 2005. [Online]. Available: http://repository.cmu.edu/cgi/viewcontent.cgi?article=1285&context=robotics&sei-redir=1.

[42] S. Welten, «Personalized Organization of Music on Mobile Devices,» ETH Zurich, Distributed Computing Group, 2009. [Online]. Available: http://disco.ethz.ch/theses/fs09/report_swelten.pdf.

[43] R. Martinec und A. Salway, «A system for image–text relations in new (and old) media (p. 370),» University of the Arts, University of Surrey, 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.9521&rep=rep1&type=pdf.

[44] J. Raphael, «Computerworld,» Computerworld, 25 February 2011. [Online]. Available: http://blogs.computerworld.com/17882/android_honeycomb_browser_user_agent.

[45] «Android Developer,» Google, [Online]. Available: http://developer.android.com/guide/webapps/best-practices.html.

8 Appendix

8.1 GetRecommendation

Element	Attribute	Description	Child element
Story (which is a child of the Recommendation element)	algorithm	Recommendation algorithm that suggested the article (int)	Comments which contains a list of comment elements and Usertitles which contains a list of usertitle elements
	articleID	The article's ID which identifies the corresponding row in the articles table (int)	
	contentID	The news provider's name (String)	
	extrVersion	Extraction instructions file version (String)	
	pubDate	The publishing timestamp formatted as: Fri, 08 Jun 2012 23:23:34 CEST (String)	
	src	The URL to the article (String)	

Table 5:GetRecommendation's Story XML

Element	Attribute	Description	Child element
Comment	commentID	The comment's ID which identifies a comment in the comments table (int)	None
	fromthisuser	Tells whether the comment is from the current user (boolean)	
	text	The comment's text (String)	
	position	The comments position within the article (int)	
	rating	The comment's rating (int)	
	numberratings	The number of times the comment was rated (int)	

Table 6: Comment element

Element	Attribute	Description	Child element
Usertitle	usertitleID	The proposed headline's ID which identifies a usertitle in the usertitles table (int)	None
	fromthisuser	Tells whether the proposed headline is from the current user (boolean)	
	text	The usertitle's text (String)	
	rating	The usertitle's rating (int)	
	numberratings	The number of times the usertitle was rated (int)	

Table 7: Usertitle element

8.2 GetTagRecommendations

Element	Attribute	Description	Child element
Story (which is a child of the TagRecommendations element)	algorithm	Recommendation algorithm that suggested the article (int)	Comments which contains a list of comment elements and Usertitles which contains a list of usertitle elements
	articleID	The article's ID which identifies the corresponding row in the articles table (int)	
	contentID	The news provider's name (String)	
	extrVersion	Extraction instructions file version (String)	
	pubDate	The publishing timestamp formatted as: Fri, 08 Jun 2012 23:23:34 CEST (String)	
	src	The URL to the article (String)	
	keyword1	First tag (String)	
	keyword2	Second tag (String)	
	keyword3	Third tag (String)	
	weight1	Weight of first tag (int 0=small or 1=medium or 2=big)	
	weight2	Weight of second tag (int)	
	weight3	Weight of third tag (int)	
	israndom	Tells whether the tag belongs to a randomly selected article (boolean)	

Table 8: GetTagRecommendations's story XML file

8.3 GetSocialData

Element	Attribute	Description	Child element
Story (which is a child of the SocialData element)	articleID	The article's ID which identifies the corresponding row in the articles table (int)	Comments which contains a list of comment elements and Usertitles which contains a list of usertitle elements
	contentID	Set to "" (String)	
	extrVersion	Set to "" (String)	
	pubDate	Set to "" (String)	
	src	Set to "" (String)	

Table 9: GetSocialData's story XML file

8.4 PostComment

Element	Attribute	Description	Child element
Comment	commentID	The comment's ID which identifies a comment in the articlecomments table (int)	None

Table 10: PostComment's Comment element

8.5 PostUsertitle

Element	Attribute	Description	Child element
Usertitle	usertitleID	The proposed headline's ID which identifies a usertitle in the usertitles table (int)	None

Table 11: PostUsertitle's Usertitle element

8.6 RateComment

Element	Attribute	Description	Child element
Comment	rating	The comment's new rating (int)	None
	numberratings	The number of times the comment was rated (int)	

Table 12: RateComment's Comment element

8.7 RateUsertitle

Element	Attribute	Description	Child element
Usertitle	rating	The usertitle's new rating (int)	None
	numberratings	The number of times the usertitle was rated (int)	

Table 13: RateUsertitle's Usertitle element

8.8 UpdateUsername and GetUsername

Element	Attribute	Description	Child element
Username	username	The user's username new rating (int)	None

Table 14: Username element of UpdateUsername and GetUsername