Effective Recommendation in Knowledge Portals

The SKYbrary case study

Roy Hoeymans
Vrije Universiteit
Amsterdam, the Netherlands
royhoeymans@gmail.com

ABSTRACT

Recommender systems are used to recommend relevant items to a user, and have been applied to several types of applications. In this paper we explore the effectiveness of a non-personalized recommender system in knowledge portals by conducting a case study on SKYbrary, a knowledge portal on aviation safety. We have developed several versions of two types of algorithms: a user-navigation based algorithm, which uses data on which articles users visit after viewing a current article, and a content based algorithm based on a vector space model with a tf-idf weighting scheme that calculates similarity between articles based on their textual content. In our first experiment, the recommendations produced by these versions are compared to recommendations that are manually selected by the content editors of SKYbrary. The best performing versions of each type of algorithm are then used to create an additional hybrid algorithm. In our second experiment, we use a survey to ask the content editors of SKYbrary to rate recommendations produced by these algorithms to determine which of the developed algorithms is the most effective. The results show that the hybrid algorithm produces significantly more relevant recommendations than a user-navigation based algorithm, but there is not enough evidence for the claim that a hybrid algorithm works better than a content based algorithm.

Keywords

Recommender system, algorithm, content based, knowledge management, knowledge portals, MediaWiki, Google Analytics, Python, tf-idf

1. INTRODUCTION

1.1 Recommender systems

A lot of websites that contain a significant amount of pages have some sort of recommender system. A recommender system is a piece of software that provides suggestions for related items to a user. This way they help users to cope with the large amount of information that is available by recommending pages, content, or products that they might be interested in [9]. Examples of this are YouTube suggesting which videos to watch, Amazon suggesting which products you might want to buy, or a news website suggesting articles that you might like to read. The object that is recommended to the user is commonly referred to as an ‘item’ [9].

There are several different classes of recommender systems based on what type of data they use [4]. One of the most widely implemented recommendation techniques is collaborative filtering, which predicts items that the current user might like based on what other users with similar preferences like [17]. Another type of recommendation technique is content based, which only looks at the current user’s historic activity and features of items; it recommends items that are similar to the items that the user has liked in the past [11]. A demographic type of recommender system recommends items based on the demographic profile of the user, for example based on the users language or age [20].

1.2 Knowledge portals

Different types of domains and situations require different types of recommendation techniques. This paper focuses on the application of a recommender system in the field of knowledge management. Knowledge management is a broad term that generally consists of the collection, organization and retrieval of information. An important part of this is developing and maintaining knowledge portals [5]. A knowledge portal is a web-based single point of access to information that is available to the public or within an organization. A well-organized knowledge portal makes it easy for its users to find and retrieve the information they are looking for, a quality that can be enhanced by the use of a recommender system. A knowledge portal can consist of web pages with text, but also databases, file systems, and interactive components such as e-learning applications or data analysis tools [5]. In this paper we focus on recommender systems in (the part of) knowledge portals that contain web pages with textual content.

1.3 Problem description

To ensure the quality of the information, the content that is available via knowledge portals is often created and maintained by domain experts. To add recommendations to a page, expert knowledge is needed, and thus the content creators often manually create and maintain a section with links to related articles. A tedious task that would ideally be supported or replaced by a recommender system. However, where websites like Amazon or YouTube have demographic information about their users and explicit information about what they like (e.g. products they have bought, ratings they gave to movies) to personalize their recommendations, knowledge portals often do not. To design a recommender system for a knowledge portal, we have to look at the types of data sources that are available. One of such data sources can be implicit user preferences [6], like how users navigate through the website and how many times a page is viewed.
Another data source that can be used is the textual content of each page. Information retrieval techniques can be used to determine how similar each page is to other pages. Recommendations can then be generated based on this similarity.

Both types of data sources have their advantages and disadvantages. The problem with the approach based on user navigation patterns is that it suffers from a 'cold start'-problem: if a page is new or unpopular, there is not enough information about how users navigate that a recommender system can use to generate useful recommendations [10]. An algorithm based on similarity between texts does not have this problem. However, such an algorithm does not take the user's preferences into account at all. In this paper, we also propose a hybrid algorithm that combines these two approaches in an effort to diminish their distinct disadvantages. We refer to the algorithm based on user navigation data as 'user-navigation based', the algorithm based on similarity between textual content of articles as 'content based', and the combination of these two algorithms as 'hybrid'.

2. RELATED WORK

2.1 Literature overview

In a literature review by Xu (2014), 403 articles that were published between 2001 and 2013 were found that are relevant to the subject of recommender systems [21]. As was mentioned in the introduction of this paper, several different types of recommendation techniques exist. Collaborative filtering is by far the most popular one, mainly because of its social value and its usefulness in the entertainment field, which is the most common application field of recommender systems [21,13]. In applications where collaborative filtering is not useful due to the lack of user data, a content based filtering algorithm is used. Hybrid algorithms, with the purpose of combining the strengths and advantages of different types of algorithms, are also common [21,22].

However, these techniques provide personalized recommendations, because data about individual user preferences and past behavior is available. The focus of this research is to find an effective way of generating recommendations for knowledge portals, which do not have access to this type of data. The recommendation techniques that we apply in this paper are thus non-personalized.

2.2 Non-personalized recommender systems

Non-personalized recommender systems based on user preferences can be divided into two main approaches. One of them is the aggregated opinion approach, which bases recommendations on the average ratings of items given by users [2]. A prerequisite for this approach is the availability of explicit ratings that users give to items, for example a score between 1 and 5. Another type of non-personalized recommender systems is a product association recommender. An example of this is Amazon's "Customers who bought this also bought". The recommendations are not specifically based on the user but on what the user is currently doing, which makes it a momentarily personalized technique [16].

The versions of the user-navigation based algorithm that we use in this paper use techniques from product association recommender systems.

In content based recommender systems, the content of items is used to predict how relevant it is to recommend them to the user based on his or her profile. The content data about an item can be structured or unstructured. Structured data consists of attributes or features. For example, a restaurant can have a 'cuisine'-attribute with possible values like 'Italian', 'French', or 'Asian' [15]. If these attributes are not available, which is often the case for web pages or documents, a content based recommender system usually relies on unstructured text. In this case techniques from the field of Information Retrieval can be utilized. An example of this is a Vector Space Model (VSM) with tf-idf weighting [9]. In a VSM, text documents and user profiles are spatially represented. The cosine similarity is then computed to predict a user's interest in a document. We also use this approach in our content based algorithm but since knowledge portals often lack user profiles we only compute the similarity between text documents and use that to generate recommendations.

While hybrid algorithms in recommender systems are common, the way we have combined a user-navigation based algorithm and a content based algorithm into a hybrid algorithm (section 6.2) is not described in the literature as of yet. The effectiveness of non-personalized recommendation has been researched in the fields of e-commerce [14] and library catalogs [19]. In both cases, non-personalized recommendations proved to be useful, be it by increasing sales or by improving search efficiency and perceived usability. In this paper the effectiveness of such a recommender system is explored in knowledge portals.

2.3 Evaluation methods

There are several evaluation methods available that can be used to determine which recommendation algorithm performs best or is most suited for the application. The most common evaluation methods are offline evaluations, online evaluations, and user studies [1]. In offline evaluations the recommendations produced by the algorithms are compared to pre-compiled recommendations in an effort to analyze how well they predict these pre-compiled recommendations. In some cases an offline evaluation is used to pre-select a set of algorithms for a more extensive evaluation [1]. In online evaluations, the behavior of users is observed when interacting with a recommender system that is already in production, for example by counting how often users click on recommended links [9]. User studies are conducted by observing test subjects while they perform tasks that require interaction with the recommender system. The results of a user study can consists of quantitative measurements like the percentage of tasks completed, the time it takes to complete a task, or qualitative answers to questions about the tasks [18].

In this paper we use an offline evaluation (section 5.4) to select two algorithms, one user navigation based and one content based algorithm, to combine into a hybrid algorithm. This hybrid algorithm is then compared to the separate user-navigation and content based algorithms by conducting a survey to domain experts (section 6.3), in which they assign a relevancy score to the recommendations produced by these algorithms. This way we acquire empirical evidence that we can use to determine which algorithm is the most effective in recommending relevant articles in knowledge portals.
3. APPROACH

3.1 Research questions

Two types of data that can be used for a recommender system are available in a knowledge portal: user navigation patterns, and the content of the articles. Respectively these types of data can be used to build a user-navigation based algorithm and a content based algorithm. The aim of this paper is to find out which of these two algorithms is more effective in a knowledge portal, and if a hybrid algorithm that combines these two algorithms to compensate for their separate disadvantages is more effective than these algorithms separately. This led us to define the main research question of this paper as follows:

- Which type of recommendation algorithm is the most effective for a knowledge portal: a user-navigation based algorithm, a content based algorithm or a hybrid algorithm that combines these two?

In order to answer the main research question, the following sub-questions need to be answered:

- Which type of user-navigation based algorithm works best for a recommender system in a knowledge portal?
- Which type of content based algorithm works best for a recommender system in a knowledge portal?
- Is a recommender system based on the combination of a content based and a user navigation based algorithm more or less effective than a recommender system based on those separate algorithms?

3.2 Scope

To answer the research question, we conduct a case study on a specific knowledge portal called SKYbrary\(^1\), a Wiki site that contains articles about aviation and air traffic management safety. SKYbrary is a good representation of a knowledge portal because it focuses on one subject, thus containing a small amount of articles compared to for example Wikipedia, and its content is created and maintained by domain experts. To ensure that the algorithms that we have developed in this paper do not only work for SKYbrary but can be used in other knowledge portals as well, we base our algorithm versions on existing techniques that are found in the literature, with little to no tweaking for SKYbrary.

SKYbrary is built with MediaWiki and uses Google Analytics to gather information about user navigation behavior. Conceptually, the algorithms that we have developed are designed to work for all knowledge portals, but the implementation of our recommender system only works for knowledge portals that are built with MediaWiki and use Google Analytics. If a knowledge portal uses another content management system, the data that is needed for the recommender system needs to be acquired in a different way.

3.3 Implementation

MediaWiki is an open source software tool that was originally developed for Wikipedia and is also used for all the other projects of the Wikimedia Foundation and numerous other Wiki sites. MediaWiki uses a MySQL database to store all the data that is relevant for a Wiki, such as data about pages, categories, links, and revisions. The raw text of each article, which is needed for a content based algorithm, can be retrieved from this database. Every knowledge portal that is built with MediaWiki has the same data structure, which makes it easy to implement our recommender system in other knowledge portals that are built with it.

Data about user navigation behavior for the user-navigation based algorithm can be retrieved via the Google Analytics API. Another method of obtaining this data is by performing log analysis on Apache server logs. However, we strongly favor Google Analytics for this task as it is simpler and more accurate; Apache logs only show requests from certain IP addresses to the server. If a user has caching enabled, most of these requests are not even contained in the log file \(^7\).

The source code\(^2\) for the recommendation algorithms we created is available online and can be used on knowledge portals that are built with MediaWiki and use Google Analytics with some modification.

3.4 Experiment setup

There are multiple ways to approach the development of a user-navigation based or a content based algorithm, which is why we have developed multiple versions of these algorithms. For pragmatic reasons, the quality of the produced recommendations can not be rated by domain experts for every version. For this reason, we conduct a first experiment in which we compare the produced recommendations to the recommendations that are manually selected by the content management team of SKYbrary. This is not the actual evaluation that we want to do but just a way of determining which versions to select for the hybrid algorithm and the second experiment.

In the second experiment we survey the domain experts to determine if the hybrid algorithm is better at recommending relevant articles than a separate user-navigation and content based algorithm. The survey shows an article and recommendations for that article, and asks the respondent to give each recommended article a score based on how relevant they think it is for a user viewing the current article. A more detailed explanation of the experiments is given in section 5.4 and 6.3.

4. SKYBRARY

We use SKYbrary for our case study on recommender systems in knowledge portals. SKYbrary is an internet platform containing articles about aviation and air traffic management safety that was launched by EUROCONTROL\(^3\), the European Organization for the Safety of Air Navigation, in 2008. The objective of SKYbrary is to be a single point of access to knowledge on these topics, and it is set up as a Wiki site so that it is publicly available online. The main audience of SKYbrary consists of people that are professionally involved with aviation or air traffic managements, like pilots and air traffic controllers.

The articles on SKYbrary can only be added or edited by a small team of aviation safety experts\(^4\) to ensure a consistently high quality of the information. In addition to writing new articles, the experts also regularly evaluate the already

\(^1\)www.skybrary.aero
\(^2\)https://github.com/Royhoey/recommender-system-knowledge-portals
\(^3\)www.eurocontrol.int/
\(^4\)www.skybrary.aero/index.php/Skybrary_Content_Management
existing articles to make sure that their content is up-to-date. This also means that the 'Related Links'-section of each article, which contains recommendations for other related articles, is reviewed and edited if needed. The recommendations that SKYbrary articles have at the bottom of each page are thus manually selected by domain experts. A recommender system that can automatically do this or support this process would be more ideal in this case. The manually selected recommendations are used to determine which algorithm version works best in our first experiment, which is described in section 5.4.

5. USER-Navigation AND CONTENT BASED ALGORITHMS

In this chapter, the several versions of the user-navigation and content based recommendation algorithms that we have developed are described in detail, and the first experiment is performed to determine which versions are used for the hybrid algorithm and the second experiment.

The recommendations are produced when selecting or visiting one SKYbrary article, which is referred to as article X. The recommended articles are referred to as articles Y. Each pair of articles X and Y has a certain score attached to it, which is used to rank the recommended articles. How this score is calculated depends on which algorithm is used.

5.1 User-navigation based

To create the user navigation based algorithm we have used the Google Analytics API to retrieve data about user navigation behavior on SKYbrary. Google Analytics only stores aggregated data and no log files, so it is not possible to retrieve information about one particular session. What can be retrieved is the total amount of browsing sessions in which a certain page is visited, and how many users visited a certain page after that. This metric is the basis for the three versions of this algorithm that we developed.

Version 1
This algorithm computes for each article pair XY in how many sessions article X was followed by article Y. This amount can be retrieved via the Google Analytics API for a certain time period. In our experiment we have used a one year period, specifically from 01-04-2014 to 01-04-2015. An example of the results of this algorithm is shown in Table 1. For example, it shows that 449 users who have visited Hypoxia, visited 'Time of Useful Consciousness' later in their browsing session. This algorithm produces accurate recommendations, mostly because the selected article X contains links to the higher ranked recommendations. The first four recommendations in Table 1 are all in the 'Related Links' section of Hypoxia, and the 'Cabin Altitude' article can be reached via a link on the page. While this algorithm is good at predicting relatedness because of this, it relies heavily on which pages are linked in the article that it produces recommendations for. Consequently these recommendations are accurate, but also quite obvious and therefore not immediately useful. However, we have used this algorithm as a base for the following more refined versions.

Version 2
The second version of the user navigation based algorithm - in addition to the XY article pair metric of version 1 - also retrieves the number of sessions in which article X is not viewed but article Y is. This number is often used in recommender system algorithms to mathematically correct for the popularity of Y by applying the following formula [16]:

\[
\frac{X \rightarrow Y}{X \rightarrow X} \times \frac{X \rightarrow Y}{X \rightarrow Y} 
\]

Use brackets to clarify what is divided by what

By doing this the so-called 'banana problem' is avoided [3]. Most of the customers in a grocery store will have a banana in their basket, because it is the most frequently bought item. Unless a formula is applied takes the popularity of the banana into account, a recommender system will almost always recommend a banana regardless of the other items that the user has in his basket, simply because it is the most popular. This algorithm looks at whether viewing article X makes viewing article Y more likely than not viewing article X, thus avoiding this problem.

Version 3
Another solution to the 'banana problem' can be obtained by applying the following formula [16]:

\[
P(X \rightarrow Y) \times P(X) \times P(Y) 
\]

What is the difference with v2? What is the intuition? Why use these two?

This produces more or less the same results as version 2 of this algorithm. Which of these two algorithms are used is determined in our first experiment (section 5.4).

5.2 Content based

For the content based algorithms we extracted the text of each SKYbrary article from the MediaWiki database. The whole body of texts were used as input for the two algorithm versions that we describe below. The output of the content based algorithms is a similarity matrix. An example of a small part of this matrix is shown in Table 2. To generate recommendations for an article X we select the row corresponding to that article in the matrix and rank the articles in the columns descendingly on similarity score. There are two versions of this algorithm, each with a different way of creating this similarity matrix.

Version 1
This version of the content based algorithm uses the term

Table 1: Top five recommendations for article X 'Hypoxia'

<table>
<thead>
<tr>
<th>Article Y</th>
<th>X → Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time of Useful Consciousness</td>
<td>449</td>
</tr>
<tr>
<td>Explosive Depressurisation</td>
<td>282</td>
</tr>
<tr>
<td>Emergency Depressurisation</td>
<td>230</td>
</tr>
<tr>
<td>Aircraft Pressurisation Systems</td>
<td>175</td>
</tr>
<tr>
<td>Cabin Altitude</td>
<td>60</td>
</tr>
</tbody>
</table>

Table 2: Example of a small part of the similarity matrix (AT = Air Temperature, DP = Dew Point, LR = Lapse Rate)

<table>
<thead>
<tr>
<th>Air Temperature</th>
<th>Dew Point</th>
<th>Lapse Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>0.46</td>
<td>0.39</td>
</tr>
<tr>
<td>DP</td>
<td>0.46</td>
<td>1.00</td>
</tr>
<tr>
<td>LR</td>
<td>0.39</td>
<td>0.41</td>
</tr>
</tbody>
</table>
frequency in the articles to calculate their similarity. An array of all the SKYbrary article texts - the corpus - is converted into an index vocabulary that contains every term present in the whole corpus, along with its frequency. Then each article in the corpus is converted to a vector of term frequencies - the measure of how many times the terms present in the index vocabulary are present in the article - and returned as a term-document matrix. The similarity between these term-frequency vectors is then calculated by computing the cosine similarity, which is a number between 0 and 1. An example of this similarity score is shown in Table 2.

Version 2
The second version of this algorithm is a more sophisticated version of the previous one. It converts the whole corpus to a term-document matrix in the same way as version 1, but also applies tf-idf (term-frequency times inverse document-frequency) normalization. Tf-idf is a term weighting scheme that is commonly used in the fields of text mining and information retrieval. It gives more importance to terms that are rare but more informative about the document than terms that are frequently found among the whole corpus [15]. For example, a word like 'pilot' gets a lower weight, because the corpus consists of aviation safety articles where 'pilot' is not a discriminative word, but a word like 'hypoxia' gets a higher weight because it is only mentioned in a few articles. If one article contains the word 'hypoxia', it is more likely to be related to an article that also contains this term, as 'hypoxia' is a term that does not occur in most articles.

5.3 Implementation
The algorithm versions that we have described were developed in Python. For the user-navigation based algorithm versions, the google-analytics-python-client5 package was used to retrieve user navigation data via the Google Analytics API. For the content based algorithm versions, scikit-learn6, a machine learning library in Python, was used. The first version of the content based algorithm uses the CountVectorizer of the scikit-learn library. This module allows us to convert an array of all the SKYbrary article texts into a spatial representation of term frequencies. The second version of this algorithm does the same, but uses the TfidfVectorizer instead, which applies the tf-idf weighting scheme. Both versions use some standard pre-processing of the texts. When converting text to a sequence of terms, scikit-learn uses the WordNgramAnalyzer, which filters out stop words, extracts tokens (e.g. "pilots" and "pilot’s" becomes the token "pilot"), removes accents and converts terms to lowercase. Additional Python packages that were used are MySQLdb7 (to store recommendations in a database), numpy8 and scipy9.

5.4 Experiment 1 - offline evaluation
In this experiment, the recommendations made by each algorithm version for each article X are compared to the articles in the 'Related Links'-section of that article, which are manually selected by the SKYbrary content editors. This se-

5https://developers.google.com/api-client-library/python/apis/analytics/v3
6http://scikit-learn.org/stable/
7https://pypi.python.org/pypi/MySQL-python
8http://www.numpy.org/
9http://www.scipy.org/

lection serves as a gold standard in this evaluation method; we compare the performance of the recommendation algorithms to the recommendations made by domain experts. For each article the top X recommendations were selected and the percentage of the recommended articles that are in the 'Related Articles'-section was calculated. For example, if an article has five related articles and the recommended articles generated by the algorithm contains two of those, the score for that article is 0.4. The average score for all articles for each algorithm version is referred to as 'recall'.

The recall is calculated for the top X recommendations for X is 2, 5, 10, and 30. Each number of X is deliberately chosen: in our second experiment we use the top two recommendations of each type of algorithm, most articles have a maximum of five recommendations in the 'Related Links'-section, we assume that a maximum of 10 articles can be somehow related, and 30 is the amount of recommendations that we generated for each algorithm version, on the assumption that a recommended article that is not in the top 30 is unlikely to be relevant.

To calculate the amount of recommended articles that match the articles in the 'Related Links'-section, we needed to know which articles are in the 'Related Links'-section. This information can not be found directly in the Mediawiki database. The text of each article was crawled until the string "== Related Articles ==" was found. In the MediaWiki language, enclosing a string with two = symbols makes it appear as a header when viewed in a web browser. Then each line was analyzed until the next section starting with two = symbols started. Each line contains a link (enclosed like "[[ link text ]]") in the MediaWiki language, which matches the page title of the linked SKYbrary article. The page title was then matched to the page table in the Mediawiki database. However, some articles are redirects. For example the link 'Crew Incapacitation' was not recognized as a related article, because it is a redirect to the article 'Pilot Incapacitation'. Because of this, the recall score refers to the amount of recommended articles generated by the algorithms that match the articles in the 'Related Links'-section that we retrieved from the text.

Not every type of SKYbrary article was included in this experiment. The only articles we included were articles that were in at least one of the three main categories: 'Operational Issues', 'Human Performance', and 'Glossary'. Articles that were excluded are articles that describe one type of aircraft, an airport, a plane crash, or safety regulation documents. The total amount of articles that are included in at least one of the main categories is 882. Of those articles, we found at least one related article in 611 of them, so ultimately this experiment was done with 611 SKYbrary articles.

5.5 Results
The recall for each algorithm version is shown in Table 3. Of all the user-navigation based algorithms, version 1 performs best. Version 1 counts the amount of sessions in which article X was followed by article Y for each article pair XY. This amount is naturally much higher for articles Y that are included in the 'Related Links'-section of article X, so the fact that this version scores higher does not mean much in this case. It does however show that the recall metric is properly calculated, because we expected this version to predict the highest amount of related articles. Version 2
Table 3: Recall of 'Related Articles' in top X recommendations

<table>
<thead>
<tr>
<th></th>
<th>X=2</th>
<th>X=5</th>
<th>X=10</th>
<th>X=30</th>
</tr>
</thead>
<tbody>
<tr>
<td>un-v1</td>
<td>0.712</td>
<td>0.758</td>
<td>0.861</td>
<td>0.940</td>
</tr>
<tr>
<td>un-v2</td>
<td>0.398</td>
<td>0.499</td>
<td>0.677</td>
<td>0.917</td>
</tr>
<tr>
<td>un-v3</td>
<td>0.397</td>
<td>0.503</td>
<td>0.669</td>
<td>0.917</td>
</tr>
<tr>
<td>cb-v1</td>
<td>0.545</td>
<td>0.590</td>
<td>0.720</td>
<td>0.838</td>
</tr>
<tr>
<td>cb-v2</td>
<td>0.605</td>
<td>0.721</td>
<td>0.880</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4: Content based recommendations for the article 'Hypoxia'

<table>
<thead>
<tr>
<th>Article Y</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency Depressurisation</td>
<td>0.48</td>
</tr>
<tr>
<td>Pressurisation Problems</td>
<td>0.38</td>
</tr>
<tr>
<td>Aircraft Pressurisation Problems</td>
<td>0.34</td>
</tr>
<tr>
<td>Cabin Altitude</td>
<td>0.34</td>
</tr>
<tr>
<td>Explosive Depressurisation</td>
<td>0.26</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Rapid Depressurisation</td>
<td>0.24</td>
</tr>
<tr>
<td>Time of Useful Consciousness</td>
<td>0.19</td>
</tr>
</tbody>
</table>

Table 5: User navigation based recommendations for the article 'Hypoxia'

<table>
<thead>
<tr>
<th>Article Y</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time of Useful Consciousness</td>
<td>1</td>
</tr>
<tr>
<td>Explosive Depressurisation</td>
<td>0.68</td>
</tr>
<tr>
<td>Emergency Depressurisation</td>
<td>0.46</td>
</tr>
<tr>
<td>Rapid Depressurisation</td>
<td>0.44</td>
</tr>
<tr>
<td>Aircraft Pressurisation Systems</td>
<td>0.38</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Cabin Altitude</td>
<td>0.23</td>
</tr>
<tr>
<td>Pressurisation Problems</td>
<td>0.13</td>
</tr>
</tbody>
</table>

Table 6: Hybrid recommendations for the article 'Hypoxia'

<table>
<thead>
<tr>
<th>Article Y</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time of Useful Consciousness</td>
<td>0.6</td>
</tr>
<tr>
<td>Emergency Depressurisation</td>
<td>0.47</td>
</tr>
<tr>
<td>Explosive Depressurisation</td>
<td>0.47</td>
</tr>
<tr>
<td>Aircraft Pressurisation Systems</td>
<td>0.36</td>
</tr>
<tr>
<td>Rapid Depressurisation</td>
<td>0.34</td>
</tr>
<tr>
<td>Cabin Altitude</td>
<td>0.27</td>
</tr>
<tr>
<td>Pressurisation Problems</td>
<td>0.26</td>
</tr>
</tbody>
</table>

6. HYBRID ALGORITHM

6.1 Purpose

The purpose of combining algorithms into a hybrid algorithm is to make up for the disadvantages that they have separately. The user-navigation based algorithm suffers from the 'cold start'-problem, which means that not enough data is present to generate recommendations for a new article because it has not been viewed yet. This can be resolved by combining it with a content based algorithm, which looks at the similarity between content and does not need data about user navigation behavior. However, a content based algorithm has several limitations as well. The content based algorithm in this paper calculates similarity between articles based on term-frequency. A disadvantage of this is that it is not possible to detect similarity between synonyms or metaphorical language [8]. However, since the SKYbrary articles contain a lot of jargon and technical terms, we think the impact of this is negligible in this case. Another drawback of content based algorithms that is often mentioned is over-specialization [9]. Because content based algorithms are so accurate they have trouble recommending something unexpected or novel to the user. This is commonly referred to as the serendipity problem [12]. While this can be a problem in fields like entertainment or e-commerce, where a recommender system is used to help the user find new items, a knowledge portal typically favors accurate recommendations over unexpected ones.

6.2 Algorithm

The intuition behind the hybrid algorithm is as follows: generate recommended articles using the content based algorithm and adjust these recommendations using its relative popularity. For example, a recommended article can have a high similarity score, but if users hardly visit it, its ranking should go down.

First we compute a set of recommended articles by using the content based algorithm. An example of a set of recommended articles is shown in Table 4. The score is the similarity score between article X (in this case 'Hypoxia') and article Y. For each recommended article, we look at the score that is assigned by the user-navigation based algorithm and 3 of the user-navigation based algorithms are two different approaches to solving the banana trap problem that we described earlier in section 5.2, with version 2 performing slightly better, which is why it is selected for the hybrid algorithm and the second experiment.

Concerning the content based algorithms, the results in Table 3 clearly show that version 2 outperforms version 1 for all sample sizes. Thus version 2 is selected for the hybrid algorithm and the second experiment as well.
### 7. RESULTS

#### 7.1 Survey results

The survey was filled out by 6 domain experts of the SKYbrary content management team. With each of them answering 15 questions that contain 2 recommendations to be rated per algorithm, a total of 150 scores for each algorithm have been gathered. The distribution of scores given for each algorithm type is displayed in Table 8.

![Table 8: Distribution of survey results per algorithm](image)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mode</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>1</td>
<td>1.27</td>
</tr>
<tr>
<td>User-navigation based</td>
<td>4</td>
<td>3.42</td>
</tr>
<tr>
<td>Content based</td>
<td>4</td>
<td>3.65</td>
</tr>
<tr>
<td>Hybrid</td>
<td>5</td>
<td>3.69</td>
</tr>
</tbody>
</table>

![Table 9: Descriptive statistics of the results of the survey](image)

In practice, most questions have less than seven recommendations because some algorithms recommend the same articles. If such an overlap takes place, the score assigned by the respondent is assigned to all the algorithms that recommend that article.

![Table 10: Results of the tests](image)

The outcome of the statistical tests shows that every algorithm performs significantly better than the baseline. It also shows that the hybrid algorithm is not significantly better at producing relevant recommendations than the content based algorithm. The difference between the content based algorithm and the hybrid algorithm is significant, as the performance of the hybrid algorithm is not different from the performance of the user-navigation-based algorithm. These results are summarized in Table 10.
Table 10: Results of Mann Whitney-U tests for algorithm pairs (UN = user-navigation based, C = content based, H = hybrid, BL = baseline)

<table>
<thead>
<tr>
<th>Algorithm pair</th>
<th>P-value</th>
<th>Significant</th>
</tr>
</thead>
<tbody>
<tr>
<td>HY - CB</td>
<td>0.516</td>
<td>No</td>
</tr>
<tr>
<td>HY - UN</td>
<td>0.02</td>
<td>Yes</td>
</tr>
<tr>
<td>HY - BL</td>
<td>0.00</td>
<td>Yes</td>
</tr>
<tr>
<td>CB - UN</td>
<td>0.09</td>
<td>No</td>
</tr>
<tr>
<td>CB - BL</td>
<td>0.00</td>
<td>Yes</td>
</tr>
<tr>
<td>UN - BL</td>
<td>0.00</td>
<td>Yes</td>
</tr>
</tbody>
</table>

and user-navigation based algorithm is not significant either. However, the hybrid algorithm is significantly better at producing relevant recommendations than the user-navigation based algorithm.

7.2 Discussion

The results show that all the algorithms perform much better than the baseline, which recommends randomly selected articles. Because knowledge portals are about one particular subject, it is not improbable that a randomly selected article is still somehow relevant to the current article. We included this baseline to see how the domain experts rated randomly selected articles compared to articles selected by our recommendation algorithms. The results show that a random article is very likely to be completely irrelevant and that the recommendation algorithms perform significantly better by a large margin.

Another observation that can be made when looking at the score distribution per algorithm in Table 8 is that the user-navigation based algorithm has a large percentage of recommendations that were rated with a score of 3, which corresponds to neutral. An explanation for this could be that users navigate to articles that the domain experts do not find relevant or not relevant enough. Therefore it would be interesting to conduct the survey with users of SKYbrary. We did not do this in our research for practical reasons; because visitors of SKYbrary are anonymous, it is hard to track them down. Also, we wanted to make sure that the recommendations were rated by experts with a similar amount of expertise on the subject of the knowledge portal, to ensure that the knowledge of the respondent on the subject does not influence the ratings they give.

Due to time constraints we did not have the opportunity to discuss the results of the survey with the respondents; an explanation of why they rated some recommendations as irrelevant might allow for more insight and can be a starting point for improvement of the algorithms in further research.

8. RECOMMENDATION ENGINE DEMO APPLICATION

There are two ways to implement a recommender system in a knowledge portal: a system where the recommenda-
Recommendation engine demo

Welcome to the recommendation engine demo. Choose a SKYbrary article that you want to show recommendations for.

Article: Hypoxia
Algorithm: Content based

Apply filters:
- Hide Accidents & Incidents articles
- Hide recommended articles that are linked in the text
- Hide recommended articles that are in the Related Links section

<table>
<thead>
<tr>
<th>Recommended article</th>
<th>URL</th>
<th>Place in article</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency Depressurisation</td>
<td>link</td>
<td>Related Links</td>
<td>0.48</td>
</tr>
<tr>
<td>Pressurisation Problems: Guidance for Controllers</td>
<td>link</td>
<td>None</td>
<td>0.38</td>
</tr>
<tr>
<td>Aircraft Pressurisation Systems</td>
<td>link</td>
<td>Related Links</td>
<td>0.24</td>
</tr>
<tr>
<td>Cabin Altitude</td>
<td>link</td>
<td>Link in article</td>
<td>0.22</td>
</tr>
<tr>
<td>Explosive Depressurisation</td>
<td>link</td>
<td>Related Links</td>
<td>0.26</td>
</tr>
<tr>
<td>Differential Pressure</td>
<td>link</td>
<td>None</td>
<td>0.26</td>
</tr>
<tr>
<td>Rapid Depressurisation</td>
<td>link</td>
<td>None</td>
<td>0.24</td>
</tr>
<tr>
<td>Pilot Incapacitation</td>
<td>link</td>
<td>None</td>
<td>0.23</td>
</tr>
<tr>
<td>High Altitude Flight Operations</td>
<td>link</td>
<td>None</td>
<td>0.19</td>
</tr>
<tr>
<td>Time of Useful Consciousness</td>
<td>link</td>
<td>Related Links</td>
<td>0.19</td>
</tr>
</tbody>
</table>

Figure 2: Screenshot of the recommendation engine demo application

recommendations are automatically generated in each article, or by using a tool to show recommendations to the content managers, who can then manually select which recommendations will be shown. This way, the content management team can still filter out possibly irrelevant recommendations. Which approach is preferred depends on what the content management team of the specific knowledge portal finds an acceptable percentage of irrelevant recommendations, and if they favor a lower workload over the quality of recommendations.

To allow for further experimentation and to provide an insight into how the different recommendation algorithms work, we have developed such a recommendation engine demo application. This tool allows the user to select an article to show recommendations for, and an algorithm to use for generating these recommendations. The tool is designed for the content management team of SKYbrary, so it is not publicly available. However, the source code\textsuperscript{10} can be adjusted to work with other knowledge portals. The tool is also useful for researchers that want to see how the recommendation algorithms behave.

Figure 2 shows a screenshot of this recommendation engine demo application. In the screenshot, the article 'Hypoxia' is selected and recommendations generated by the content based algorithm are shown in the table below. The table consists of the title of the recommended article, a link to that article on SKYbrary, the relation of the recommended article to the current article (it can be linked in the article or already included in the 'Related Links'-section), and the score assigned by the algorithm.

A few extra filters were also added specifically for SKYbrary. The 'Hide Accidents & Incidents articles' filter hides recommended articles that are in the Accidents & Incidents category. Articles that are from that category are never recommended, because each article has a separate section containing links to related accidents and incidents. The other two filters can be used to hide recommended articles that are linked in the article text or recommended articles that are already included in the 'Related Links'-section. These filters allow the content managers of SKYbrary to quickly see if they have missed any relevant articles that are not linked or included in the 'Related Links'-section in the article.

\textsuperscript{10}https://github.com/Royhoey/Recommendation-engine-demo-application
9. CONCLUSION

9.1 Conclusion

The main research question of this paper, which we have stated in section 3.1, is: ‘Which type of recommendation algorithms is the most effective for a knowledge portal: a user-navigation based algorithm, a content based algorithm or a hybrid algorithm that combines these two?’ The first experiment showed us which version of each algorithm is better at predicting pre-selected related articles. This way we could answer our sub questions and combine the best user-navigation based version and the best content based version into a hybrid algorithm. The purpose of the hybrid algorithm is to compensate for the disadvantages of the separate algorithms, and is thus expected to produce more relevant recommendations. To test this we conducted a survey to the content management team of SKYbrary, in which they were asked to rate the recommendations generated by the different algorithms on relevancy.

The results of this survey show that there is enough evidence to say that a hybrid algorithm works better than a user-navigation based algorithm. However, we can not say with absolute certainty that a hybrid algorithm works better than a content based algorithm, or that the content based algorithm works better than the user-navigation based algorithm. When implementing a recommender system in a knowledge base, a hybrid algorithm that also factors in content similarity by using a tf-idf weighting scheme should be considered over an algorithm based on user navigation behavior. The results of our survey show that a hybrid algorithm is likely to be better than a content based algorithm, but a statistically significant difference in performance could not be proven. To draw this conclusion, a different or more elaborate evaluation method might be needed.

A recommendation engine demo application was designed to allow researchers and content managers of knowledge portals to experiment with recommendation algorithms. This way a decision can be made on which algorithm to implement, or the tool can be used to support the process of manually selecting recommended articles.

9.2 Discussion and future work

We have used several existing algorithms to generate recommendations for a knowledge portal in this paper, while purposely not tweaking the algorithms for our case study to ensure that they will work for other knowledge portals as well. Studies on different knowledge portals with the same algorithms can add support to this claim of generalizability.

A limitation of the user-navigation based algorithm in this case study is that the SKYbrary articles already contain a 'Related Links'-section. An article that might be relevant but is not linked in that section will never achieve a high rank in the set of recommendations. Therefore it would be interesting to experiment with knowledge portals that do not have a 'Related Links'-section yet and see how a user navigation based algorithm works there.

The results of the survey show that the difference in performance between the content based algorithm and the hybrid algorithm is subtle, which made it difficult to find a statistically significant difference. This can be caused by the small amount of respondents of the survey we conducted. This can be an inducement for further research on a knowledge portal where a higher amount of respondents is available to rate the algorithms. An evaluation technique that can be used is A/B testing, where half of the visitors of the knowledge portal see recommendations made by algorithm A and the other half sees recommendations made by algorithm B. This will lead to a much larger amount of data points. Another cause of this subtle difference in performance could be the way the user-navigation and content based algorithms were combined into a hybrid algorithm. Different ways of combining these algorithms into a hybrid algorithm can be explored and evaluated in further research as well. These starting points for future work might allow for a more evident conclusion on which recommendation algorithm is the most suitable for knowledge portals.

10. REFERENCES

[13] Deuk Hee Park, Il Young Choi, Hyea Kyeong Kim, and Jae Kyeong Kim. A review and classification of recommender systems research. School of


