Where will you comment next?
Exploiting comments for personalized recommendations

P.S.N. Chandrasekaran Ayyanathan
WHERE WILL YOU COMMENT NEXT?

EXPLOITING COMMENTS FOR PERSONALIZED RECOMMENDATIONS

by

P.S.N. Chandrasekaran Ayyanathan

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science

at the Delft University of Technology,
to be defended publicly on Wednesday October 21, 2015 09:30 AM

Supervisor: Dr. M. A. Larson
Thesis committee: Prof. dr. ir. A. Hanjalic, TU Delft
Dr. ir. C. Hauff, TU Delft

This thesis is confidential and cannot be made public until 21 October 2015.

An electronic version of this thesis is available at http://repository.tudelft.nl/.
ABSTRACT

Since the advent of Web 2.0, users have not only increasingly created content, but also contributed reactions to content in the form of comments. Comments are challenging to analyze due to their short lengths and informal style, meaning that any individual comment provides very little data to work with and is highly variable. However, comments capture innate and an explicit opinion of a user that makes it invaluable towards personalization. In this work, we explore the possibilities of exploiting comments towards the end of personalized recommendations.

Over the course of this work, we investigate the particular domain of news recommendation and report our findings through use of different recommenders evaluated offline. Our contributions include an evaluation strategy that allows for simulation of recommenders offline, a simplistic hybrid filtering technique that exploits the advantages of its root recommenders and various findings related to news recommendation in general.

We perform a preliminary study into investigating whether users maybe attributed by comments they make and find that they are indeed attributable if the right features are considered. Utilizing the property of authorship attribution through comments, we achieve user-user similarity that ultimately aids in delivering recommendations. We find that freshness is an important aspect in news recommendation and therefore for the design of our recommender we build upon the freshness aspect while also achieving personalization by exploiting content, user-user similarity and the user neighbourhood.
ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Dr. Martha Larson for her continued support in the duration of my thesis. She was the person to introduce me to the phenomenal field of recommendation and helped me realize my passion towards this ever growing field.

I express my gratitude towards Ir. Gijs Koot and Dr. Lucia D’Acunto of TNO for giving me the opportunity to work at a distinguished organization such as TNO and allowing me to work on their dataset. I appreciate the support provided by fellow TNO colleagues and mentors throughout my internship.

Many thanks to Dr. Claudia Hauff and Prof. Alan Hanjalic for being part of the committee and providing valuable feedback to improve my work. My heartfelt thanks to all friends, family and peers from the MMC department who gave me inspiration as well as support.

P.S.N. Chandrasekaran Ayyanathan
Delft, October 21, 2015
# Contents

1 Introduction .......................................................... 1  
  1.1 Research Flow ................................................. 1  
  1.2 Comments as Feedback in Recommendation .................. 2  
  1.3 Thesis Organization ......................................... 3  

2 Background and Related Work .................................. 5  
  2.1 Authorship Attribution ....................................... 5  
  2.2 Recommender Systems ....................................... 6  
    2.2.1 Collaborative Filtering ................................. 6  
    2.2.2 Content Based Filtering ............................... 6  
    2.2.3 Hybrid Filtering ...................................... 7  
  2.3 Recommendation Using Comments ............................ 7  

3 Dataset ..................................................................... 9  
  3.1 Newsfeed of NU.nl ............................................. 10  
  3.2 Statistics ......................................................... 10  
    3.2.1 Publishing and Commenting Behaviour ............... 10  
    3.2.2 Item Based Statistics .................................. 13  
    3.2.3 User Based Statistics .................................. 16  

4 Attribution of Comments ........................................ 19  
  4.1 k-NN Distance Based Attribution ............................ 19  
  4.2 Attribution Experiments ...................................... 21  
  4.3 Inferences ......................................................... 25  

5 Evaluation and Framework ....................................... 27  
  5.1 Sampling .......................................................... 27  
  5.2 Framework ......................................................... 29  

6 Recommendation Algorithms ................................... 31  
  6.1 Existing Recommenders of NU.nl ............................ 31  
    6.1.1 Freshness Recommender ................................. 31  
    6.1.2 Popularity@n Recommender ............................. 32  
  6.2 Content Recommender ........................................ 32  
    6.2.1 Tags ........................................................ 33  
    6.2.2 Article .................................................... 33  
    6.2.3 Comments ................................................ 34  
  6.3 SVD++ Recommender .......................................... 35  
  6.4 Weighted Reciprocal Rank Fusion Recommender ........... 36  

7 Results and Discussion ........................................... 39  
  7.1 Results .......................................................... 39  
  7.2 Conclusion and Outlook ....................................... 43  

A Appendix .............................................................. 45  
  A.1 NU.nl Layout .................................................... 45  
  A.2 Terms and Formulae ........................................... 47  
    A.2.1 TF*Idf Weights .......................................... 47  
    A.2.2 Distance Measures ...................................... 47  
    A.2.3 Matrix Factorization .................................... 47  
    A.2.4 Weighted Reciprocal Rank Fusion ....................... 48  
    A.2.5 Quantile Bounded Mean ................................. 48  
    A.2.6 Metrics ..................................................... 48
Bibliography
Most of us would have had the experience of visiting our favorite restaurant where due to our frequent dining the waiter knows our personal preferences and tastes. Even without our explicit mention, the waiter recommends dishes taking our preferences and personal tastes into consideration.

Recommender systems are no less different from that waiter catering to our personal preferences. They strive to satisfy the user’s needs by providing a quality experience taking his tastes and interests into account. Human intelligence has the flexibility to easily be applied in a range of situations. In contrast recommender systems must be designed with specific objectives in mind and are heavily influenced by the target domain under consideration.

In this work, we investigate the particular domain of news recommendation. In contrast to other domains, news has the innate property of being relevant only if it is fresh and recent, users are not so much interested in legacy news items and are more interested in the current affairs. As a result, we face ever evolving datasets and it is a challenging task to keep track of this rapidly changing user interest.

The novelty of our work lies in exploiting comments and their content as a form of feedback to assess this rapidly changing interest. Comments do not often receive the credit they deserves as the common notion is that they are too short and often noisy to be of any use. In this work, we bust this myth by showing comments are useful in their own way.

1.1. Research Flow

Over the course of our work, we will answer the research questions illustrated in Figure 1.1. The sections where the questions are answered are indicated by the floating numbers over the blocks.
1.2. COMMENTS AS FEEDBACK IN RECOMMENDATION

Recommender systems heavily rely upon user feedback as a form of learning user behaviour and interests. Conventional recommender systems make use of explicit user ratings on items and where ratings are not available, they make use of clicks. For further personalization, clicks are often enriched with additional meta-data relating to the item or the user.

Various forms of meta-data have been extensively studied as forms of user feedback. Ye et al. investigate geo-location awareness enriched upon social networking for recommending new locations [1]. Castagnos et al. report implications of eye tracking for recommendation in an e-commerce scenario [2]. Liu et al. move further into the user’s body by investigating user’s heartbeat as an attribute towards personalized music recommendation [3]. While the rate at which sensors evolve over the years, it is inevitable that this would be the future of recommendations. But, there is a more personal and explicit form of feedback that has been ignored all along, namely comments.

What are innate characteristics of comments that make them unique from other forms of feedback?

Our hypothesis is that comments explicitly express the particular elements in a topic the user would be interested in. In this work, we perform exploratory research into discovering the usefulness of this unique user feedback medium towards personalization in recommendations.

Consider Figure 1.2 where a NU.nl article about the ‘Nationaal Hitteplan’ regarding the Dutch government prescribing a policy to follow for the hot weather. Comment ‘11’ by a user mentions that in Germany and Belgium the weather was much cooler, comment ‘12’ makes a remark on the suitedness of this weather for Ramadan and comment ‘15’ mentions that it is a weather for delicious ice creams. While these comments do not necessarily relate to the Dutch hot weather policy specifically, they explicitly states what each user was interested in when it comes to hot weather.
1.3. Thesis Organization

Chapter 2 provides a brief background on recommender systems and work relating to our own study. We perform an in-depth analysis of the data we investigate for our recommendation algorithm in Chapter 3. The adopted evaluation strategy and framework are discussed in Chapter 5.

As we are interested in exploiting comments as a form of feedback, we explore whether users are distinguishable on the basis of their comments by investigating authorship attribution in Chapter 4. Basing upon our findings relating to the dataset and our authorship attribution results, we discuss the different recommender algorithms we propose in Chapter 6 and finally discuss the results in Chapter 7.
BACKGROUND AND RELATED WORK

In this chapter, we first provide a brief background on authorship attribution and research in this area that relate to our work, then we provide an introduction to recommender systems and state-of-art in the area of recommender systems research related our work.

2.1. AUTHORSHIP ATTRIBUTION

We earlier mentioned that we would be looking into authorship attribution in Chapter 4, if users are indeed attributable to their own comments, then we would be able to use their commenting vocabulary as a means of discerning the collaborative pattern of users while also accounting for the explicit content expressed in comments.

Authorship attribution aims at predicting the writer of a given a piece of text. In general a variety of stylistic and lexical features are used. Authorship attribution problems generally arise in forensic settings, and involve relatively long documents, and lists of possible authors limited to a few hundred candidate authors. Often, a sizable amount of example text is available for each candidate. In contrast, in our use case, the documents are short, the list of candidate authors is very large, and most are associated with little example text.

Stamatatos provides an excellent summary of existing authorship attribution techniques [4]. Most of the work discussed make use of both large documents and a smaller candidate author set. Stamatatos also points out that Support Vector Machines (SVM) perform extremely well as a distinguishing methodology in this particular scenario.

More recent work that looks into a smaller documents was done by Layton et al. [5], who look into authorship attribution for Twitter, where a micro-post (i.e., tweet) is restricted to 140 characters. Layton et al. opt for profile based feature representation of documents wherein all documents of a particular user are concatenated to form a super-document. They experiment with character n-grams with varying n values and find that 4-grams provide overall a good mean accuracy.

Work on large-scale attribution was carried out by Narayanan et al. [6], who predict authors from a list of 100,000 candidate authors using a range of classifiers. Their dataset was comprised of blog posts, which are relatively longer than tweets or comments. They employed parts-of-speech tagging and a variety of lexical/character features for their purposes. A very interesting find in their work was that simple classifiers such as Naive Bayes and k-Nearest Neighbour performed better than SVM at the top ranks.

Our use case deals with shorter documents, namely comments, and a large candidate author set. We experiment with parts of methodologies adopted by Layton et al. [5] and Narayanan et al. [6] as our problem is a combination of both. Next, we turn to the discussion of recommender systems.
2.2. **RECOMMENDER SYSTEMS**

Recommenders aim to predict a user's interest towards a web item by utilizing a variety of data ranging from the user's behaviour history to the actual content of the web item. Recommender systems have become a very popular phenomenon and are prevalent in the web, be it electronic retailers such as eBay and Amazon or social networking sites such as Facebook and Twitter, the underlying motivation across domains remains relatively the same – to retain the user on the site by delivering him content he would find interesting.

Earlier we discussed on how different forms of feedback affect recommendations. Now, we briefly discuss the major techniques employed in recommender systems.

2.2.1. **COLLABORATIVE FILTERING**

Collaborative filtering heavily relies on feedback provided by other users in addition to the target user. It operates on the neighbourhood of users and exploits shared opinions to make recommendations.

Some of the first work that investigated collaborative filtering was by Goldberg et al. [7] in an electronic mail scenario wherein they recommend mailing lists for users to follow. Their proposed system accepted user annotations to documents, these annotations are then shown to other users who filter through documents using these crowd-sourced annotations. These annotations form a feedback mechanism and allow users to manually reach documents that interest them.

The work of Goldberg et al. was more directed towards crowd-sourced tagging to aid in search tasks. A full fledged recommender was proposed by Resnick et al. [8] who recommend news articles for users to read. The system accepted explicit ratings from users and correlated users with one another to make rating predictions for new items the user has not rated yet.

Since then, hardware and computation costs have significantly changed allowing for more sophisticated algorithms to be built for making more accurate rating predictions. One example would be the Gravity recommendation engine by Takács et al. [9] which utilizes matrix factorization through gradient descent optimization for rating predictions. Matrix factorization in itself has also been enhanced for different scenarios – Koren [10] proposed a new model that is better suited for implicit feedback scenarios, Manzato et al. [11] introduced a further modified model that allows for utilizing the contextual features as dimensions in the factorization model and so many more [12–14].

No matter the sophistication of the underlying the algorithm, the core concept remains the same – exploit feedback given by other users in order to make a recommendation to the target user. The only drawback in collaborative filtering is its inability to produce recommendations for item/user cold start scenarios wherein there are no feedback received/given by the item/user.

2.2.2. **CONTENT BASED FILTERING**

Content based filtering utilizes the content of the item for making recommendations. This content is usually in the form of text and ranges from tags to article descriptions to additive meta-data. The content maybe machine generated or manually written, therefore relies on the origin and relevancy of content to make accurate predictions.

Foltz and Dumais [15] proposed one of the first content based filtering techniques in a technical memo recommendation scenario. The memos accepted feedback from employees on a scale of 1 to 7 to represent their interest towards that particular memo. Using these ratings the recommendation engine then filters through new memos by keyword search and latent semantic indexing.

A more widely known content based filtering application among the research community is CiteSeer by Bollacker et al. which aids in retrieving scientific publications [16]. CiteSeer maintains a parsed representation of documents which are then retrieved for a user query. A specialty of CiteSeer is the ability to display similar documents, this is done through $tf \times idf$ vectors. In recent years, CiteSeer has evolved into CiteSeerx [17].
Content based filtering therefore creates a profile of the user based upon the items he has expressed feedback towards, the profile is then used to generate recommendations. In recent years, content based filtering does not act on its own in a recommendation scenario and is often bundled together with collaborative filtering techniques thereby creating a hybrid recommendation engine.

2.2.3. **Hybrid Filtering**

Hybrid filtering aims to use the abilities of both collaborative and content based filtering to produce personalized recommendations that take into consideration both the user neighbourhood and the content.

One of the first hybrid implementations was by Lang [18] who designed an engine that recommended news items. The meta recommender essentially fuses the scores provided by different recommenders through weighted averages. More recently, Saveski and Mantrach [19] proposed a matrix factorization model that utilizes local collective embeddings, which essentially combines collaborative and content matrices into a common space. This common space is then used for predicting ratings and consequently generate recommendations.

In summary, hybrid filtering aims to eliminate the shortcomings of both collaborative and content-based filtering techniques by combining both into a seamless singular engine that provides better recommendations.

2.3. **Recommendation Using Comments**

Comments have long been under the radar of research, some notable works include the very detailed study of weblog comments by Mishne and Glance [20], summarization of blogs through the comments they receive by Hu et al. [21], popularity prediction of news articles utilizing comments they receive by Tatar et al. [22], search re-ranking of images through comments by San Pedro et al. [23] and many more.

In a recommendation scenario, comments are not as widely studied. One of the prominent contributions was by Shmueli et al. [24]. They utilize article content, tags assigned to items and the user interactions as implicit feedback to form a latent space which undergoes matrix factorization. The latent factor parameters then undergo gradient descent optimization so as to minimize a cost function. A special characteristic is that they utilize the target item which the user commented on as positive feedback and a different item the user did not comment on as negative feedback in order to calculate the cost function. But, a caveat of this calculation is that we cannot be fully sure that an item a user did not comment on would qualify as negative feedback, it might just as well be that the user might have liked the item but did not come across the item to make a comment on it in the first place.

Saveski and Mantrach [19] improved upon the work of Shmueli et al. Their underlying methodology remains creates a common latent space for both content and user interactions as well, except instead of matrix factorization they propose collective factorization that exploits the locality of points in the latent space. Their methodology also allows for explainable recommendations that produce fine grained topical explanations.

Both works of Shmueli et al. and Saveski and Mantrach do not exploit the content of the comments and view comments as merely another form of implicit feedback. One of contributions that also utilize the content of comments was by Li et al. [25] where they constructed language models from comments on news articles and used these models to subsequently retrieve articles for recommendation. The key motivation in their case was that comments convey the evolving topics of news articles and thereby allowing for trend detection. But, their technique lacked personalization and viewed comments only as a means for discerning topic evolution.

A more recent work was by Bansal et al. [26]. Their work improved upon that of Li et al. by also relating the comment content with specific topics of the article. This relational topic profile is then representative of each user and is used to deliver personalized recommendations. They also account for factors such as article popularity and existing comments for further personalization. The target domain of their recommender was generic and was not tailored towards a particular domain. Our own work exclusively focuses on news and therefore the domain heavily influences our design as we will be discussing in the later Chapters.
The dataset under observation is NU.nl, a popular Dutch online news website that comprises of predominantly Dutch comments scraped from their commenting platform NUjij.nl and spans over the three months of October 2014 - December 2014. The particular period of 3 months was chosen so as to accommodate for a uniform distribution of articles and comments. News pages during earlier periods were not indexed properly as a result of which scraping them did not provide us with the actual content.
An article is initially published on the original NU.nl site by editors, this is then shared by users and thereby receives comments through NUjij.nl. Therefore, the user can come across the original article through both NU.nl or NUjij.nl but eventually the comment is made only in NUjij.nl. Figure 3.1 shows the overall layout of NU.nl and how users receive recommendations, Figures A.1, A.2 and A.3 in Appendix A.1 show a breakdown of the interface in both NU.nl and NUjij.nl.

3.1. NEWSFEED OF NU.NL

NU.nl has mainly three main newsfeeds in place – these can be considered non-personalized recommendation systems of NU.nl,

1. Feature - Editor curated lists displayed on the homepage and updated periodically.
2. Freshness - The most recent articles are recommended to read/comment.
3. Popularity - The most clicked recent articles are recommended to read/comment.

The feature articles are both fresh and also related to a trending topic. In addition they receive the most coverage in the user interface as seen in Figure 3.1. As we opt for offline evaluation, it will be a difficult task to out-perform for a system recommending these feature articles due to their quality of being both fresh and trendy. But, there are always cases where the user might have discovered and eventually commented on articles which are not featured on the homepage. In this thesis, we are interested in individual personal behaviour of users, although the clicks recorded in our dataset can be assumed to represent it only implicitly.

To summarize, NU.nl or NUjij.nl do not have any form of personalized recommender in place that utilizes user preferences and hence the dataset would also be generally influenced by these generic freshness/popularity based recommenders.

3.2. STATISTICS

Table 3.1 shows a few brief statistics of the dataset, we present more detailed statistics related to the dataset in the upcoming sections.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Users</td>
<td>26,791</td>
</tr>
<tr>
<td>No. Items</td>
<td>18,874</td>
</tr>
<tr>
<td>No. Commented Items</td>
<td>10,833</td>
</tr>
<tr>
<td>No. Comments</td>
<td>372,454</td>
</tr>
</tbody>
</table>

Table 3.1: Overall Statistics of NU.nl

These simple statistics inform us on what aspects of the data to address and will be discussed in the upcoming sections.

3.2.1. PUBLISHING AND COMMENTING BEHAVIOUR

In this section, we will be looking into a few statistics of the publishing and commenting behaviour at NU.nl. The publishing statistics encompass all items irrespective of whether they received comments whereas the commenting statistics encompass the comments.

Figure 3.2 shows the number of articles published and number of comments made in each hour of the day. As can be seen the active publishing hours are from 07:00 - 17:00 while the very early hours do not see as much publication. This correlates with the number of comments made, though the number of comments seem to decline at a slower pace in the later hours compared to the decline in the number of publications.
### 3.2. Statistics

#### (a) With Respect to Items

![Hour Based Items](image1)

**Figure 3.2: Hour Based Statistics**

#### (b) With Respect to Comments

![Hour Based Comments](image2)

**Figure 3.3: Hour Based Statistics**

Figure 3.3 shows the number of articles published and number of comments made in each day of the month, the X-axis runs from Monday through Sunday. The weekends see the least number of publications while the weekdays are more or less uniform. Yet again this behaviour correlates with the number of items though there are relatively more comments being made in the weekends.

#### (a) With Respect to Items

![Day Based Items](image3)

**Figure 3.3: Week Based Statistics**

#### (b) With Respect to Comments

![Day Based Comments](image4)

**Figure 3.3: Week Based Statistics**

Figure 3.4 shows the number of articles published and number of comments made in each month of the entire dataset from October 2014-December 2014. For both number of comments and number of publications made, the behaviour is uniform though there is a relatively higher number of comments made in December even though that is the least published month.
(a) With Respect to Items

Figure 3.4: Month Based Statistics

Earlier we looked into the raw number of comments and publications across different timer periods (hour, day and month), now we look into the average number of publications and comments across the same time periods in Table 3.2.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Publications</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. in an Hour</td>
<td>8.55</td>
<td>168.68</td>
</tr>
<tr>
<td>Avg. in a Day</td>
<td>205.15</td>
<td>4048.41</td>
</tr>
<tr>
<td>Avg. in a Month</td>
<td>6291.33</td>
<td>124151.33</td>
</tr>
</tbody>
</table>

Table 3.2: Statistics Related to Publishing and Commenting

Table 3.3 shows the number of articles published in each section for the top 10 sections whereas Table 3.4 shows the number of comments each section received for the top 10 most commented sections. Inferring from the table, the sections that receive more articles do not necessarily receive the most comments.

<table>
<thead>
<tr>
<th>Section</th>
<th>Article Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>buitenland</td>
<td>1583</td>
</tr>
<tr>
<td>binnenland</td>
<td>1219</td>
</tr>
<tr>
<td>sport</td>
<td>1054</td>
</tr>
<tr>
<td>achterklap</td>
<td>964</td>
</tr>
<tr>
<td>politiek</td>
<td>775</td>
</tr>
<tr>
<td>economie</td>
<td>549</td>
</tr>
<tr>
<td>internet</td>
<td>478</td>
</tr>
<tr>
<td>muziek</td>
<td>443</td>
</tr>
<tr>
<td>film</td>
<td>411</td>
</tr>
<tr>
<td>beurs</td>
<td>409</td>
</tr>
</tbody>
</table>

Table 3.3: Top 10 Section-Wise Article Count

<table>
<thead>
<tr>
<th>Section</th>
<th>Comment Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>binnenland</td>
<td>65790</td>
</tr>
<tr>
<td>buitenland</td>
<td>61657</td>
</tr>
<tr>
<td>politiek</td>
<td>59205</td>
</tr>
<tr>
<td>economie</td>
<td>24459</td>
</tr>
<tr>
<td>achterklap</td>
<td>23045</td>
</tr>
<tr>
<td>sport</td>
<td>22419</td>
</tr>
<tr>
<td>internet</td>
<td>13456</td>
</tr>
<tr>
<td>gadgets</td>
<td>10962</td>
</tr>
<tr>
<td>geldzaken</td>
<td>10769</td>
</tr>
<tr>
<td>ondernemen</td>
<td>9262</td>
</tr>
</tbody>
</table>

Table 3.4: Top 10 Section-Wise Comment Count

For instance, the category 'gadgets' did not even occupy the top 10 of most published categories but is one of the top 10 most commented categories. Articles related to 'gadgets' mainly triggered comments about users not liking a particular company’s products. This goes on to say that the article content matters as well for it to be deemed comment-worthy for users.
3.2.2. **Item Based Statistics**

This section relates items to the number of comments they receive and, for this reason is restricted to only those items that received comments. Table 3.5 shows statistics related to items receiving comments.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. No. Comments</td>
<td>1.0</td>
</tr>
<tr>
<td>Max. No. Comments</td>
<td>1112.0</td>
</tr>
<tr>
<td>Mean No. Comments</td>
<td>34.38</td>
</tr>
<tr>
<td>Median No. Comments</td>
<td>17.0</td>
</tr>
<tr>
<td>Var. No. Comments</td>
<td>2781.28</td>
</tr>
</tbody>
</table>

Table 3.5: Comment Statistics Per Item

There is a very large variance indicating that the distribution of comments across items is highly uneven. The mean is twice the median which points to the fact that the number of comments are not evenly distributed across items. The top 5 articles with most comments are tabulated in Table 3.6.

<table>
<thead>
<tr>
<th>News Article</th>
<th>Comment Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negentig aanhoudingen bij aankomst Sinterklaas in Gouda</td>
<td>1112</td>
</tr>
<tr>
<td>‘Kaas- en stroopwafelpieten’ bij landelijke Sinterklaasintocht</td>
<td>873</td>
</tr>
<tr>
<td>Kabinet probeert crisis af te wenden</td>
<td>849</td>
</tr>
<tr>
<td>Forse kritiek op truc coalitie om zorgplannen door te zetten</td>
<td>800</td>
</tr>
<tr>
<td>Albert Heijn haalt Zwarte Piet uit reclame-uitingen</td>
<td>638</td>
</tr>
</tbody>
</table>

Table 3.6: Most Commented Articles in the Dataset

The top most commented articles are related to a very controversial topic specific to Dutch culture and stems from a Dutch holiday known as ‘Sinterklaasfeest’ which is celebrated during the early days of December. The third and fourth most commented articles relate to politics specifically to the ‘Gesneuvelde Zorgwet’ regarding a dispute in agreement of health care in Netherlands occurred again during the month of December. These articles would have no doubt triggered users to comment and voice their own opinions about the topics.

We draw a few random articles that received no comments during the period under our observation and tabulate them in Table 3.7.

<table>
<thead>
<tr>
<th>News Article</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zweedse overname Fins staalbedrijf</td>
</tr>
<tr>
<td>‘Sociaal netwerk Path bespreekt overname door berichtenapp KakaoTalk’</td>
</tr>
<tr>
<td>Van Gaal hoop dat overtuigende zege United ‘boost’ geeft</td>
</tr>
<tr>
<td>Zes tips voor meer veiligheid op de werkvloer</td>
</tr>
</tbody>
</table>

Table 3.7: Articles that Received No Comments

The least commented articles more or less deal with passive topics and do not necessarily stimulate heated debates amongst the users. It also depends on how these articles were recommended to the users, if they were not shown on the homepage of the website then it is less likely that users would have come across the article in the first place.

Figure 3.5 is a histogram of the number of items in each range of comment count bin. Each bin represents collection of items comprising of comments in the corresponding specified range. As can be seen, a large portion of commented items have only 1-100 comments.
For a meaningful interpretation, we drill down the 1-100 range in Figure 3.6 which is a histogram of the number of items with less than 100 comments in each comment count bin. Here, there is more gradual decrease compared to the previous abrupt fall in Figure 3.5.

We drill further down to the 1-10 range in Figure 3.7 which is a histogram of the number of items with less than 10 comments in each comment count bin. The trend is more or less uniform with items having 2 comments being the most numerous.
Table 3.8 shows a few temporal statistics that would further shed light on the dataset. We discard a few odd extremely high values as they are not representative of the rest of the dataset. For instance, Figure 3.8 shows a comment that occurred after a full 20 months of article publication, this seems to have been caused by a push notification the user received after the article was updated after a year of its publication. This was the only comment on that article as a result of which including this in our temporal statistics would heavily offset our observations.

$T_f$ represents the time difference between article publication and the first comment while $T_l$ represents the time difference between article publication and the last comment.
### Table 3.8: Temporal Statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value in Seconds</th>
<th>Value in Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. $T_f$</td>
<td>23</td>
<td>0.0064</td>
</tr>
<tr>
<td>Max. $T_f$</td>
<td>223611</td>
<td>62.11</td>
</tr>
<tr>
<td>Mean $T_f$</td>
<td>5368.7075</td>
<td>1.49</td>
</tr>
<tr>
<td>Min. $T_l$</td>
<td>78</td>
<td>0.0217</td>
</tr>
<tr>
<td>Max. $T_l$</td>
<td>350018</td>
<td>97.23</td>
</tr>
<tr>
<td>Mean $T_l$</td>
<td>101688.4738</td>
<td>28.24</td>
</tr>
<tr>
<td>Min. $T_l - T_f$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max. $T_l - T_f$</td>
<td>337354</td>
<td>93.71</td>
</tr>
<tr>
<td>Mean $T_l - T_f$</td>
<td>93523.58</td>
<td>25.98</td>
</tr>
</tbody>
</table>

The mean time to last comment sheds light into the duration up till which articles are considered fresh. Users at NU.nl on average comment on only the recent 28 hour articles - this is yet again influenced by the existing recommenders. Therefore, we must also for our own recommender use this particular period as a means for querying items to recommend. The mean time between first and last comments essentially represents the average lifetime of an article, the period is close to that of the mean time to last comment.

**3.2.3. User Based Statistics**

In this section, we relate users who made comments in the 3-month period of our dataset to the number of comments they make. Table 3.9 shows statistics relating users and comments.

### Table 3.9: Comment Statistics Per User

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. No. Comments</td>
<td>1.0</td>
</tr>
<tr>
<td>Max. No. Comments</td>
<td>907.0</td>
</tr>
<tr>
<td>Mean No. Comments</td>
<td>13.90</td>
</tr>
<tr>
<td>Median No. Comments</td>
<td>3.0</td>
</tr>
<tr>
<td>Var. No. Comments</td>
<td>1210.52</td>
</tr>
</tbody>
</table>

As with the items, the variance is large and the mean is higher than the median, implying that the distribution of comments made by users is highly uneven.

Figure 3.9 is a histogram of the number of users in each comment range. Similar to the items, there is an abrupt fall and most users only made as many as 1-100 comments.
3.2. STATISTICS

Figure 3.9: Histogram of No. Users vs Comment Count Ranges

We drill down to the 1-100 range in Figure 3.10. The behaviour is still somewhat similar to the previous observation but not as abrupt.

Figure 3.10: Histogram of No. Users < 100 Comments vs Comment Count Ranges

Finally, we look at the 1-10 range in Figure 3.11. There is a more gradual decrease but also goes on to point out that most users only made 1 comment in the entire dataset.

Figure 3.11: Histogram of No. Users < 10 Comments vs Comment Count Ranges
Modelling around users who only made a single comment would not be very meaningful. We also assume that users who made 2 comments would also not be very helpful towards creating personalized recommendations. We utilize users who made at least 3 comments for all our experiments. Taking a higher threshold would always yield richer representations but going any higher than 3 would possibly ignore a majority of the user base.
We aim to eventually utilize the content of comments for recommendation, therefore we must first understand the characteristics of comments. The authorship attribution study in this chapter aims at seeking an understanding of the strength of the association between comments and their authors. This would provide insight into user commenting behaviour, in particular into whether a user has a unique vocabulary and stylistic pattern that distinguishes him from other users. If the user does indeed have a distinguishing commenting characteristic, we can utilize the content of comments as form of a collaborative pattern instead of the user himself.

### 4.1. k-NN Distance Based Attribution

In our scenario, the dataset is highly imbalanced, with a vast majority of authors having made very few comments and a handful of authors who made hundreds of comments. For this reason, we adopt instance-based classification, wherein each document contributes on its own to the attribution, and choose a simple 1-NN classifier.

For a given comment, we predict the nearest most resembling comment in the training set. It is to be noted that we do not make profile based predictions by grouping all comments of a single user together but rather each comment constitutes its own class as we previously established there is an uneven distribution in the number of comments per user. The overall framework is illustrated in Figure 4.1.
Figure 4.1: Authorship Attribution

The features we make use of are the character $n$-grams similar to those of Layton et al. [5], since our use case also deals with similar informal text and shorter documents. We experiment with different values of $n$. We perform basic thresholding by selecting only those $n$-grams that occur more than five times in the entire dataset. This thresholding is not per user or comment but on the $n$-gram vocabulary of the entire dataset so as to eliminate odd $n$-grams and thereby reduce the feature size upon we operate.

The character $n$-grams are converted into $tf*idf$ weight vectors, $tf*idf$ weights are essentially the product of the term frequency in a single document and the inverse document frequency. They aid in distinguishing the unique terms in a document, and directly expose user-specific features. $tf*idf$ is generally formulated as,

$$tfidf = f_{t,d} \cdot \log \frac{N}{n_t}$$

where, $f_{t,d}$ - frequency of term $t$ in document $d$

$N$ - total documents in corpus

$n_t$ - number of documents with term $t$

Each vector represents a single comment. If a particular $n$-gram is not present then the corresponding $tf*idf$ entry is zero in that index. We account for user specific vocabulary by utilizing $n$-grams as features and also the distribution/frequency of these $n$-grams that is characteristic of the target user that makes him distinguishable among other users through the $tf*idf$ vectors.

We experiment with two different distance functions as a means for finding their nearest neighbours. Namely, euclidean distance similar to what Narayanan et al. [6] used and cosine similarity. If $p_i$ and $q_i$ are two vectors then we formulate,

$$\text{Euclidean Distance} = \| (q - p) \|$$

$$\text{Cosine Similarity} = \frac{p \cdot q}{\|p\| \cdot \|q\|}$$

where $\|x\| = \sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_n^2}$ is the norm of the vector $x$
Euclidean distance directly measures the magnitude of the distance between the two vectors whereas cosine similarity measures the directional difference between the two vectors.

### 4.2. Attribution Experiments

We wish to make predictions based on historical data as in any real world scenario, one can only predict future behaviour based on past instances and not the other way around. Therefore, we split the data into rigid train and test sets – the first 10 days of comments are taken as the training set and the next 3 days as the test set. These 10-3 day splits are then slid over the entire dataset, resulting in total of 79 train-test splits of the data. Figure 4.2 illustrates the sampling strategy.

![Figure 4.2: Sampling Strategy for Authorship Attribution](image)

Figure 4.2: Sampling Strategy for Authorship Attribution

We ensure that all users who occur in the testing set occur in training set as well since it would not be possible to predict for new users. We also ensure that the test users. Table 4.1 shows a few statistics in the underlying split of the data averaged over all the samples. The exact 10-3 days split was chosen as approximations so as to account for sufficient train and test samples.

<table>
<thead>
<tr>
<th>Train/Test</th>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>Mean No. Users</td>
<td>2529.49</td>
</tr>
<tr>
<td></td>
<td>Mean No. Items</td>
<td>1267.42</td>
</tr>
<tr>
<td></td>
<td>Mean No. Comments</td>
<td>25217.55</td>
</tr>
<tr>
<td></td>
<td>Mean No. Neighbourhood Size</td>
<td>460.89</td>
</tr>
<tr>
<td>Test</td>
<td>Mean No. Users</td>
<td>2518.58</td>
</tr>
<tr>
<td></td>
<td>Mean No. Items</td>
<td>461.07</td>
</tr>
<tr>
<td></td>
<td>Mean No. Comments</td>
<td>8302.33</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics referring to Adopted Sampling Strategy

The metric we use for evaluation are MRR and MAP@10 – the definitions are given in Appendix A.2.6. We calculated a trivial baseline that drew a comment from the training set at random while prioritizing users who made more comments. The baseline had an $MRR < 0.001$ in both cases of NU.

Table 4.2 shows the results of classification averaged over all comments.

<table>
<thead>
<tr>
<th>Distance Measure</th>
<th>n-gram</th>
<th>MRR</th>
<th>MAP@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>2</td>
<td>0.0283</td>
<td>0.0144</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.0305</td>
<td>0.0163</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.0245</td>
<td>0.0135</td>
</tr>
<tr>
<td>Cosine</td>
<td>2</td>
<td>0.0664</td>
<td>0.0356</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td><strong>0.0711</strong></td>
<td><strong>0.0412</strong></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.0562</td>
<td>0.0302</td>
</tr>
</tbody>
</table>

Table 4.2: Per-Comment Results
Clearly, using 3-grams with cosine similarity based prediction outperforms any of the Euclidean distance based predictions. This may be due to the fact that cosine similarity outright ignores vectors which are entirely dissimilar whereas the euclidean distance measure does not. In the case of text, dissimilar vectors directly imply two pieces of text with entirely different words. This better suits our goal of trying to find similar comments.

We note that, it is open to question whether authorship attribution utilizing cosine similarity for all domains would outperform Euclidean distance. Narayanan et al. [6] had used Euclidean distance for the special case of blogs which are much larger than comments and also a different set of features.

Can users be attributed based upon their commenting behaviour?

Considering the scale of results for our classification task wherein there were potentially around 4k different users to attribute to, we were able to make a prediction on average in the top 15 ranks. This implies that specific users of NU.nl more or less utilize the same words over time that makes them uniquely identifiable.

If one were to make use of conventional authorship features such as text compression, parts-of-speech, lexical features (frequency of punctuation, word length, sentence length), which capture subtle stylometric features such as those employed by Narayanan et al. [6] rather than just character n-grams, one might see an improvement over the base performance. But, use of such features is out of scope in this work.

In the end we were still able to make decent predictions even without such advanced features and were able to assess that the average user of NU.nl utilizes similar vocabulary over time. We also look into the stability of the predictions over different conditions so as to determine whether it is biased towards a particular nature of the dataset.

All plots are essentially histograms averaged with respect to the particular X-axis metric. Due to the wide variations that would potentially occur due to the wide range in the actual number of instances in each bin, we utilize Quantile Bounded Mean (QBM) as an approximation towards the trend of the plot.

To calculate the QBM approximation, we split our dataset that is binned with respect to the X-axis metric then split into 5% quantiles. We then find the average over these quantiles resulting in a single data point for that quantile, connecting all such data points finally results in a smoothed approximation of the trend of the plot. We do not take into account the last quantile as the scaling of X-axis values would not allow us to discern the trend.

Figure 4.3: MRR Predictions vs Bins of No. Training Comments per User Averaged over All Splits
4.2 Attribution Experiments

We initially look at how the average number of training instances that were used per user in all the train-test splits impacted the MRR score. Figure 4.3 is a plot of the MRR predictions versus the bins of average number of training comments that were utilized. As can be seen most of the instances occurred from 3 - 7 training comments, there is a slight fall between 3 and 4 but it picks up after 4 and generally increases from there on. The fall is characteristic of how users who made less than 4 comments do not necessarily comment with the similar vocabulary - but with more comments a more discernible pattern is observed thereby resulting in higher MRR scores.

![Figure 4.3: MRR Predictions vs Bins of No. Unique Users who Commented over Entire Dataset](image)

Next, we look at whether the number of comments on the item over the entire dataset affected our predictions. Figure 4.4 is a plot of the MRR predictions versus the bins of number of unique users who had commented on the item in the entire dataset. This measure is essentially a surrogate of the popularity of the item - the more the unique users who commented the more popular the item is expected to be. There is a slight increase as the number of unique users increases - with a significant rise from 1 - 2 unique users and generally it stabilizes after 7 with slight increases towards the end.

![Figure 4.4: MRR Predictions vs Bins of Neighbourhood Size of User Averaged over All Splits](image)

Next, we look at whether the number of comments on the item over the entire dataset affected our predictions. Figure 4.4 is a plot of the MRR predictions versus the bins of number of unique users who had commented on the item in the entire dataset. This measure is essentially a surrogate of the popularity of the item - the more the unique users who commented the more popular the item is expected to be. There is a slight increase as the number of unique users increases - with a significant rise from 1 - 2 unique users and generally it stabilizes after 7 with slight increases towards the end.

![Figure 4.5: MRR Predictions vs Bins of Neighbourhood Size of User Averaged over All Splits](image)

Figure 4.5 is a plot of the MRR predictions versus the bins of average user neighbourhood sizes across the splits. The user neighbourhood is simply computed by the sum of the unique users who co-commented
along with the target user. The neighbourhood size is then averaged over all splits for each user and thereby binned to represent in the plot. An increase in the average neighbourhood size could also be said to be an increase in the number of comments made by the user - therefore we observe a similar trend as that of Figure 4.3. The MRR stabilizes after a neighbourhood size of 30. However, there are no drastic changes.

We further investigate the co-commenter behaviour by utilizing the target user’s neighbourhood users’ labels as the ground truth rather than only the target user’s label. The intention is to study whether users who comment on similar items use similar vocabulary. We utilize cosine distance for our predictions as we had previously observed that cosine distance performed better. Table 4.3 shows the results of the co-commenter classification.

<table>
<thead>
<tr>
<th>Predictor</th>
<th>MRR</th>
<th>MAP@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-NN Cosine</td>
<td>0.3472</td>
<td>0.3293</td>
</tr>
<tr>
<td>Random</td>
<td>0.2150</td>
<td>0.2068</td>
</tr>
</tbody>
</table>

Table 4.3: Per-Comment Co-Commenter Results

A surprising observation was that random predictions perform close to that of our k-NN predictions. This is characteristic of the neighbourhood size - as we previously in Table 4.1 observed that the average neighbourhood size was 460.89 per user, considering this with the average number of users as seen in Table 4.1, we can safely state that it is only a matter of predicting from a small handful candidate clusters for a correct prediction.

As before, we investigate particular properties of the dataset and their influence on our predictions.

![Figure 4.6: Co-Commenter MRR Predictions vs No. Training Comments](image)

(a) Cosine k-NN
(b) Random

Figure 4.6 is a plot of the co-commenter MRR predictions versus the number of training comments that were utilized. Both cosine k-NN and random predictions observe the same trend of gradual increase as the number of comments increases, since increase in number of comments potentially increases the number of users upon whom we evaluate i.e. increase in size of the ground truth labels under consideration.
4.3. **Inferences**

In conclusion, we were able to make decent predictions for single label attribution even with simple features. The accuracy of the single label attribution also encourages us to model the user by his unique comment content. Thereby allowing for a more rich representation of the user while also taking into account user-user similarity based upon their commenting vocabulary.

In addition, for full fledged authorship attribution, one might employ the technique used here as a sampling strategy to narrow down a large number of authors to a candidate set that then later undergoes the post-processes utilizing stylistic features for accurate prediction.

The co-commenter prediction shows the innate nature of the dataset wherein the same set of news items are recommended to all users resulting in users interacting with almost the same set of items over time. But,

---

**Figure 4.7**: Co-Commenter MRR Predictions vs No. Unique Users who Commented

Figure 4.7 is a plot of the MRR predictions versus the number of unique users who had commented on the item. Only earlier on, there is a slight rise and a subsequent fall whereas the rest is stable. This might be an indication that users with very large neighbourhood sizes comment on the least popular articles thereby causing a slight rise whereas the medium popular that ranges from 7 are commented by the average user.

**Figure 4.8**: Co-Commenter MRR Predictions vs Neighbourhood Size of User

Figure 4.5 is a plot of the MRR predictions versus the user neighbourhood sizes. This without a doubt shows the anticipated, that as the neighbourhood size increases, the number of ground truth labels increase and subsequently the MRR increases.
considering the fact that most users only make very few comments as observed in Section 3.2.3, it is not as straightforward as to implement a uniform recommender that recommends the same items to all users.
In this chapter, we briefly discuss the evaluation strategy adopted and the framework in which the recommender algorithms introduced are plugged into.

5.1. Sampling

Due to the infeasibility of deploying a live recommender on NU.nl, we opt for offline evaluation that simulates the online environment. In an online scenario, a recommendation is generated every time the user logs in or whenever a user views a news article. These log-in or viewing meta-data are absent in our dataset, therefore we assume that the time at which the target user commented is the time at which he received a recommendation. We refer to the time a target timepoint. Our recommender problem is to predict an item for each target timepoint.

In a live recommendation scenario, we can only make through training historic data. In our case as well, whenever a comment is made (implicitly recommendation received) we look back in the immediate 10 days to retrieve any other comments the target user made and treat them as training instances. A lower bound for number of training instances was chosen as 3 inferring from our observations in Figure 3.10 as going any lower would not produce meaningful patterns. Therefore, we do not take into consideration users who did not make at least 3 comments in the immediate 10 days for recommendation. Table 5.1 shows a few statistics relating to the comments that satisfy these conditions.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Comments</td>
<td>82702</td>
</tr>
<tr>
<td>No. Users</td>
<td>6545</td>
</tr>
<tr>
<td>No. Items</td>
<td>5532</td>
</tr>
<tr>
<td>Mean No. Training Instances</td>
<td>17.62</td>
</tr>
</tbody>
</table>

Table 5.1: Statistics Relating to Comment-Wise Sampling

Now that we have our training instances upon which we train our model, the next task is to retrieve candidate items to make a recommendation for the target comment. Since we are recommending news articles, freshness comes into the picture and we do not wish to recommend legacy articles which are out of date. Therefore, initially we retrieve all news items irrespective of whether they received comments in the immediate 48 hours and remove items which the target user has already commented on. These final fresh news items are treated as candidates for recommendation. The specific period of 48 hours of candidate items was chosen higher than what was observed in Table 3.8 where the average time to last comment was approximately 28 hours. Figure 5.1 illustrates the sampling strategy.

5
5. EVALUATION AND FRAMEWORK

(a) Select Time Point T

Retrieve all Interactions for Training

(b) Retrieve Training Instances

Retrieve Items not commented by Target User as Candidates

(c) Retrieve Candidate Items for Recommendation

Figure 5.1: Comment-Wise Sampling
5.2. Framework

How to evaluate a comment centric recommender offline?

Due to the infeasibility of deploying a live recommender, we aim for a simulated offline evaluation through our framework. The framework relies entirely on the discussed sampling strategy, given a target comment, we would predict the corresponding target item from the recent 2 days of items through the knowledge of past 10 days of comments. Different recommenders are plugged into the framework in order to rank the recent 2 days of items, the ranking is subsequently evaluated against the target item.

![Overall Framework](image)

We impose a few conditions in the creation of the recommender:

- A target comment is chosen as a target timepoint only if the target user has made 3 comments in the past 10 days of the comment's timestamp. This is because we believe going any lower would not produce meaningful recommendations and going any higher would discard a large user base.

- The candidate items for recommendation are only items that have been published within the last 48 hours due to the fact that the dataset is more inclined towards fresh items and we also had observed previously in Table 3.8 that the average time to the last comment on an item was close to 28.

- Candidate items which have already been commented on by the target user are removed from the candidate list since we do not want to recommend items that have already been commented on by the user.

- The recommendation only predicts the single item which the target comment corresponds, even if the target user had commented on a recommended item in the future but not at the time instant we are looking for, we penalize the recommender for a miss.
In this chapter, we discuss the proposed methodology by describing the different algorithms that are plugged into the framework we discussed earlier.

6.1. EXISTING RECOMMENDERS OF NU.nl
As we opt for offline evaluation of our recommender, it is vital to understand the existing recommender in place as the dataset would be influenced by this recommender. As we previously established in Section 3.1, NU.nl there are three explicit recommenders namely - editor picked features, freshness based and popularity based. Since, it is not an easy task to imitate editor picked features without knowing what the actual entries were during that time, we will only be imitating approximately by using our versions of the freshness and popularity based recommenders.

6.1.1. FRESHNESS RECOMMENDER
The freshness based recommender is a simplistic recommender that produces a ranked list of the most recent news items that were published. The ranking is based on the recency of the article - the articles which are closer to the current timestamp occupy the top ranks.

Figure 6.1 illustrates the overall framework - it is to be noted that \( T-n \) represent the time point preceding \( n \) hours before the target time point \( T \).

![Figure 6.1: Freshness Recommender Design](image-url)
6.1.2. **Popularity@N Recommender**
The popularity based recommender is an enhanced version of the freshness based recommender wherein a ranked list of recent items that have received more clicks is produced. Initially, the most recent items with respect to a particular time period are retrieved and then these items are ranked based upon the number of clicks they received.

In this case, we limit the number of items we retrieve to particular time periods $n$. Items after $n$ are then ranked based upon the decreasing number of comments they received until the target timepoint. An assumption we make here is that the number of clicks is proportional to the number of comments, as to comment on an article in the first place the user must have clicked on it. For items which have not received comments till the target timepoint, we utilize the freshness recommender as fallback and rank them by their recency. These zero comment recency based ranked items are then appended to the end of the original popularity based ranks. Figure 6.2 illustrates the overall framework. In the final candidate list of items - item $A$ has the most items whereas item $Z$ has the least.

![Figure 6.2: Popularity Recommender Framework](image)

6.2. **Content Recommender**
In a content recommendation scenario, as we had previously discussed in Section 2.2.2, we aim to make predictions through the content of the item. This is usually in the form of text and utilizes various metadata to constitute a profile of the item used to make predictions.

We adopt the same features of $tf \times idf$ vectors as we had used in Chapter 4 but since this is not a classification problem and more of a recommendation one, therefore we slightly modify the strategy. The overall methodology is somewhat similar to that of what Shepitsen et al. [27] discuss in their content recommender where they essentially generate clusters through different techniques for candidate items through the $tf \times idf$ vectors then consequently for a recommendation, they would find the most similar cluster through cosine similarity and make a recommendation.

In our case,

1. Initially retrieve only items which the target user has commented on in the immediately preceding 10 days and add them to Set $S_i$.
2. Retrieve all items which the user has not commented on in the immediately preceding 2 days and add them to Set $S_j$.
3. Pre-process all items of both Sets $S_i$ and $S_j$. The pre-processing strategy differs based upon the underlying data that is used.
4. Generate $tf \times idf$ vectors for all pre-processed tokens in both Sets $S_i$ and $S_j$. 
5. Find the average vector $V_x$ of the $tf*idf$ vectors of all items in Set $S_i$. Consider $V_i$ to be the $tf*idf$ vectors of items in Set $S_i$ then,

$$V_x = \frac{1}{n} \sum_{i=1}^{n} V_i$$

6. Compute cosine similarity of $tf*idf$ vectors of each item $j$ in Set $S_j$ to average vector $V_x$. Consider $V_j$ to be the $tf*idf$ vectors of an item in Set $S_j$ then cosine similarity $sim_j$ of item $j$ is given by,

$$sim_j = \frac{V_j \cdot V_x}{||V_j|| \cdot ||V_x||}$$

7. Produce a ranked list of items in the increasing order of $sim_j$ which is then recommended to the target user.

![Content Recommender Design](image)

Figure 6.3 illustrates the afore mentioned process. We essentially construct a user profile based upon the items the target user commented on in the past by computing $tf*idf$ vector of the content of the commented items and averaging over all the commented item vectors. The content differs with respect to the underlying data (Tags, Article or Comments). Candidate items are then compared with this user profile and ranked based upon their vector space similarity. The implicit assumption here is that the user would be interested in commenting on items with similar content. The following sections describe the different underlying data we experimented with.

### 6.2.1. Tags

NU.nl articles are published along with tags that are manually assigned by editors and are related to what the article is about. For the purposes of our recommender we treat each tag as a singular token after which the $tf*idf$ vectors are directly constructed without any additional pre-processing steps. On average there were 1.72 tags per article and 45 articles did not possess any tags at all. In the cases where there no tags we do not construct $tf*idf$ vectors but instead fall back to the freshness recommender and append these tag-less articles to the end of the computed ranked list.

### 6.2.2. Article

Here we use the actual article text written by the editors for recommendation. In this case, we tokenize through white spaces, tokens are then subjected to a snowball stemmer [28] and finally common Dutch words are removed thereby leaving tokens which represent the article. The pre-processed tokens are then used to
construct the $tf \times idf$ vectors.

Many articles possess inline HTML tags that are embedded videos or URLs to other web pages, we experiment both article content with HTML and without HTML. Tables 6.1 and 6.2 show the most informative terms found when considering content with HTML and without HTML. The article-wise $tf\times idf$ weights were first found, then these weights were averaged across articles belonging to the same section and sorted by their weight.

<table>
<thead>
<tr>
<th>Section</th>
<th>Top Tf*Idf Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>binnenland</td>
<td>politie, man, brand, verdacht, brandweer, uur, onderzoek, h2</td>
</tr>
<tr>
<td>buitenland</td>
<td>oekraïn, land, mens, president, reger, militair, volgen, irak, politie, russisch</td>
</tr>
<tr>
<td>politiek</td>
<td>kamer, minister, pvda, kabinet, vvd, partij, d66, nederland</td>
</tr>
<tr>
<td>economie</td>
<td>procent, bank, gripenland, economie, euro, europees, h2, griek, miljard</td>
</tr>
<tr>
<td>achterklap</td>
<td>em, a, actrie, relatie, vertelt, jarig, zanger, kinder, acteur, zangers</td>
</tr>
<tr>
<td>sport</td>
<td>span, <a href="http://www.nusport.nl">www.nusport.nl</a>, wk, voetbal, a, wedstrijd, 1, seizoen, nbsp, we</td>
</tr>
<tr>
<td>internet</td>
<td>googl, app, gebruiker, a, facebook, internet, uber, nbsp, appl, bedrijf</td>
</tr>
<tr>
<td>gadgets</td>
<td>appl, android, a, <a href="http://www.nutech.nl">www.nutech.nl</a>, samsung, app, googl, window, scherm, iphon</td>
</tr>
<tr>
<td>geldzaken</td>
<td>procent, euro, <a href="http://www.nugeld.nl">www.nugeld.nl</a>, consument, bank, gemiddeld, pensioen, h2, hypothek, a</td>
</tr>
<tr>
<td>ondernemen</td>
<td>procent, bedrijf, euro, ondernem, span, miljoen, nederland, bedrijf, jar, h2</td>
</tr>
</tbody>
</table>

Table 6.1: Top 10 Section-Wise Most Informative Tf*Idf Terms for Articles with HTML.

In some cases of utilizing HTML in construction of $tf\times idf$ vectors, HTML tag elements are added to the mix. In the case of the ‘sport’ section, it contains many HTML elements as top terms which goes on to point out that articles in ‘sport’ section contain more embedded elements such as videos, slides, tables, etc. Including HTML elements therefore also aids in distinguishing articles from one another and also captures the stylistic patterns adopted by the editors themselves. We also experiment with a variant where we append the tag tokens to our pre-processed article tokens and then construct $tf \times idf$ vectors.

6.2.3. Comments
We assume that comments are a form of crowd-sourced tags which describe the interesting elements in the article, specifically those which were found to be interesting by users. In addition, we also hypothesize that they could possibly produce a collaborative pattern considering our observations in Chapter 4.

We only retrieve all the comments made from the start of the training period up till our test time point and ignore all other comments. As comments are often prone to spelling mistakes, the comments are subjected to character $n$-gram tokenization. We choose $n=3$ for tokenization as we previously observed in Chapter 4 that they work best for attributing NU.nl comments. These character $n$-gram tokens of all comments to a particular article are grouped together to form a super document which is then used to construct a $tf \times idf$
vector. For a singular item, its \( tf \cdot idf \) vector represents all its comments. For candidate items which had not received comments in the training period, we fall back to the freshness recommender and append these articles without comments to the end of the computed ranked list.

6.3. **SVD++ Recommender**

The SVD++ recommender is essentially an implementation of the modified matrix factorization model proposed by Koren [10]. We aim to exploit any collaborative pattern that could be observed by utilizing the existence of a comment as a form of feedback.

Matrix factorization is factorization taken to the matrix level. Initially, a user-item ratings matrix \( r_{ui} \) is constructed with respect to the interactions,

\[
\begin{pmatrix}
    r_{1,1} & r_{1,2} & \cdots & r_{1,i} \\
    r_{2,1} & r_{2,2} & \cdots & r_{2,i} \\
    \vdots & \vdots & \ddots & \vdots \\
    r_{u,1} & r_{u,2} & \cdots & r_{u,i}
\end{pmatrix}
\]

Then, through a singular value decomposition technique such as gradient descent optimization the matrix factors \( q_i \) which represents items and \( p_u \) which represents user are computed,

\[
\hat{r}_{ui} = b_{ui} + q_i^T \cdot p_u
\]

This direct factorization works in scenarios where there are explicit ratings for instance on a scale of 1 to 10 but what we are dealing with here is the existence of the comment i.e. implicit feedback. Koren proposed a modified version of the matrix factorization to better suit implicit feedback scenarios. Here a new matrix \( y_j \) is introduced, this represents items towards which the target user expressed implicit feedback. The final SVD++ model is given as,

\[
\hat{r}_{ui} = b_{ui} + q_i^T \cdot \left( p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)
\]

where,

- \( b_{ui} \) - user-item bias
- \( N(u) \) - set of items user \( u \) expressed implicit feedback towards
- \( y_j \) - implicit feedback vector for item \( j \)

Our SVD++ recommender was designed as follows,

1. Retrieve all comments in the immediate 10 days and generate a user-item matrix \( r_{ui} \) that has an entry of 1 if a user commented on an item and an entry of 0 if a user did not comment on an item.
2. Initialize \( b_{ui}, q_i, p_u \) and \( y_j \) matrices to random values drawn form a Gaussian distribution in \([-1,1]\).
3. Compute \( \hat{r}_{ui} \) as specified in the SVD++ model and find the error \( e \),

\[
e = r_{ui} - \hat{r}_{ui}
\]

4. Update all parameters when error \( e \) is not minimal,
where,  
\[ b_{ui} = b_{ui} + \gamma \ast (e \ast q_i - \lambda \ast b_{ui}) \]  
\[ p_u = p_u + \gamma \ast (e \ast q_i - \lambda \ast p_u) \]  
\[ q_i = q_i + \gamma \ast (e \ast \left( p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) - \lambda \ast q_i) \]  
\[ y_j = y_j + \gamma \ast (e \ast |N(u)|^{-\frac{1}{2}} \ast q_i - \lambda \ast y_j) \]  

5. Repeat Steps 3 and 4 until there is minimal observable error \( e \) for all interactions. In our case, we repeated the step for 20 iterations.

6. Retrieve all items in the immediate 48 hours and compute the estimated rating \( \hat{r}_{u,i} \) the target user would give to those candidate items.

7. Sort the estimated candidate item ratings in descending order to retrieve a ranked list of items.

Figure 6.4 illustrates the process defined above. Matrix factorization also requires specifying the number of dimensions to be specified for the factor matrices. These dimensions are considered to be features of the user/item. We adopted a dimension/feature size of 20 for our experiment as was employed by Shmueli et al. [24] in their scenario.

![Figure 6.4: SVD++ Recommender Design](image)

Candidate items which had not received comments in the training period, we fall back to the freshness recommender and append these articles without comments to the end of the computed ranked list.

6.4. **Weighted Reciprocal Rank Fusion Recommender**

Now that we have accounted for all possible forms of feedback and designed different recommenders that utilize these different forms of feedback, the final step is to exploit all these recommenders to produce a more refined ranked list that takes into account the innate properties of the different recommenders through a singular hybrid recommender.

We design our hybrid recommender out to be a re-ranker that takes the ranks of different recommenders as input and re-ranks them to produce the optimal result. There are numerous ranked list combination techniques prominent of which are CombMNZ by Fox et al. [29], Condorcet Fusion by Montague et al. [30] and more recently Reciprocal Rank Fusion by Cormack et al. [31].
CombMNZ utilizes the explicit scores given by different rankers whereas Condorcet Fusion is similar to that of majority voting. The explicit scores given by our rankers are very different from one another and it does not make sense to use these disjoint values. Cormack et al. propose a simple method that utilizes the reciprocal rank and found that this method outperforms Condorcet Fusion.

For an instance \( i \) ranked by different rankers \( r \in R \) reciprocal rank fusion is given as,

\[
RRF(i) = \sum_{r \in R} \frac{1}{k + r(i)}
\]

where \( k \) is a constant

We introduce weighted reciprocal rank fusion where \( k \) is variable and is modified based upon the current recommender. This way we prioritize the reciprocal ranks produced by specific recommenders over others.

\[
WRRF(i) = \sum_{r \in R} \frac{1}{k_r + r(i)}
\]

where \( k_r \) - recommender priority

The smaller the value of \( k_r \) the higher the priority assigned to the corresponding recommender would be and vice versa. The priority can be understood as a weight. For instance, if \( k_f = 10 \) for the freshness recommender and \( k_t = 20 \) for the content recommender, we prioritize the ranks provided by the freshness recommender over the content recommender. The process is illustrated in Figure 6.5.

![Figure 6.5: WRRF Recommender Design](image-url)
In this chapter, we report the findings related to the recommenders we briefly discussed in the previous chapter and also pave the way for future research.

7.1. Results

Table 7.1 tabulates the results of all previously discussed recommenders. For the evaluation of each recommender, we use the metrics MRR, P@1 and MAP@10, the definitions of which can be found in Appendix A.2.6. We limit the MAP score to rank 10 as we assume that in a live recommender scenario the user is not provided with more than 10 results at a time.

<table>
<thead>
<tr>
<th>Recommender</th>
<th>MRR</th>
<th>P@1</th>
<th>MAP@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freshness</td>
<td>0.09786</td>
<td>0.0221</td>
<td>0.0739</td>
</tr>
<tr>
<td>Popularity@48</td>
<td>0.0564</td>
<td>0.0167</td>
<td>0.0405</td>
</tr>
<tr>
<td>Article(Non-HTML)</td>
<td>0.0658</td>
<td>0.0231</td>
<td>0.0515</td>
</tr>
<tr>
<td>Article(HTML)</td>
<td>0.0829</td>
<td>0.0360</td>
<td>0.0686</td>
</tr>
<tr>
<td>Tag</td>
<td>0.0038</td>
<td>0.0017</td>
<td>0.0037</td>
</tr>
<tr>
<td>Tag+Article(HTML)</td>
<td>0.0833</td>
<td>0.0358</td>
<td>0.0690</td>
</tr>
<tr>
<td>Comment</td>
<td>0.0734</td>
<td>0.0281</td>
<td>0.0566</td>
</tr>
<tr>
<td>SVD++</td>
<td>0.0647</td>
<td>0.0485</td>
<td>0.0485</td>
</tr>
<tr>
<td>WRRF(^1)</td>
<td>0.2201</td>
<td>0.1201</td>
<td>0.1998</td>
</tr>
</tbody>
</table>

Table 7.1: Results of All Recommenders

\(^1\) - Priority (k): Freshness = 30, SVD++ = 40, Tag+Content = Comment = 70

Does freshness/trend matter?

The freshness recommender is the best performing individual recommender, which is a clear indication of the property of news and its inclination towards freshness. Inferring from the MRR, on average we were able to make correct predictions within the top 10 ranks. As this is a mere recency predictor, it translates that users usually comment on the 10 most recent articles. Comparing this to the observations in Table 3.2 we could even say that the average user comments on items published in the recent 1.5 hours.

As for the popularity@\(n\) recommender, we do not know the exact look back time period up till which items are to be retrieved and treated as fresh, we experiment with different values of \(n\) in the range 1-48 hours from the target timestamp. Figure 7.1 shows a plot of the predicted MRR scores against the different look back timestamps adopted.
Figure 7.1: MRR Predictions of Popularity vs Look Back Timestamps

As the look back timestamp increases, the MRR score decreases - this maybe partly due to the reason that as we increase the look back time we also increase the number of candidate items. But, the MRR more or less stabilizes after 30 hours and does not have steep drops as in the earlier regions. This also partly attributes to the lifetime of articles as seen in Table 3.8 - after close to 30 hours most articles reach the end of their lifetime and would have the same number of comments over time which would consequently result in the same list of recommendations irrespective of whether it goes as far 48 hours.

While popularity@1 may seem like a very good predictor, it is to be noted that it works on a very small candidate set. Recall from Table 3.2 that in an hour on average close to only 9 articles are published, in contrast to popularity@48, which works on more than 400 articles. We demonstrate this in Figure 7.2 where we plot the number of zero MRR scores against the look back time. As can be seen, in spite of having a good MRR score, popularity@1 has missed out on almost 40k interactions. Therefore for future reference and for a fair comparison, we would only use popularity@48.

Figure 7.2: Histogram of No. Zero MRR Scores vs Look Back Timestamps

The other individual recommenders could not out-perform the freshness recommender, which clearly affirms our theory that the dataset is inclined towards fresh items. The tag-based recommender performs the worst, considering the fact that there were on average only 1.72 tags per article, it is not sufficient to provide
7.1. **Results**

recommendations.

**Does the neighbourhood matter?**

The SVD++ recommender performs poorly as a singular recommender reflecting the fact that there is very little collaboration among the users of NU.nl. In other words, users do not necessarily share a common commenting pattern. In addition a majority of users only make very few comments that makes discerning a clear collaborative pattern a difficulty.

**Does content matter?**

The recommender with article content with HTML tags performs the best as a singular recommender other than freshness, adding tags to the mix slightly increases the performance over article content without tags. But, when HTML tags are removed from the article, the performance drops which is an indication that users not only comment on articles with similar article content but also articles which possess similar HTML tags. As we consider $tf*idf$ vectors as features, it is possible that the HTML tags would have generated unique features for each article in the case of articles. For instance, articles which refer to the same URLs would be considered very similar to one another.

The comment content recommender outperforms the article without HTML recommender and is also a close contender to the article with HTML recommender. This points to the conclusion that comments possess as much valuable information as article content. Considering that we utilized character $n$-grams which capture stylistic behaviour, also strengthens the fact that users commenting on similar items have similar stylistic characteristics.

**Do different features combined together matter?**

We are interested in determining the upper bound for the combination of features. Therefore we set the parameters of the hybrid recommender by permuting over a selected set of parameters ranging from 10-70. The exact limits of 10 and 70 were chosen so as to not offset the ranks to a very large degree. The parameters were permuted till the highest MRR score was obtained. We also experimented with a scenario where all recommenders were given equal priority and observed that it was able to still out-perform all individual recommenders. Therefore, the hybrid recommender brings out the best out of each recommender.

We investigate with respect to how the recommender fairs against the different conditions with which NU.nl recommends, namely against popularity and freshness.

Figure 7.3 was created by computing the MRR across items and plotting them against the bins of the number of unique user comments the item received during the entire duration. We then find the $QB M$ up till the 99th quantile for a clearer observation of the trend. Here we observe how the different recommenders play against items of different popularity.
Freshness out-performs all singular recommenders and also out-performs the hybrid recommender for items with a smaller number of comments. The freshness recommender slightly rises and is more or less stable after around 10 comments whereas all other recommenders gradually increase. The trend of the hybrid recommender is characteristic towards this behavior, as all single recommenders perform worse for items with fewer comments, so does the hybrid recommender but unlike being stabilized as in freshness, it increases towards the end under the influence of the content recommenders.

The rate of increase in performance is highest with the comment content recommender as it is directly dependent on the comments and an increase would increase its feature size. The SVD++ recommender performs very close to the popularity recommender while being very slightly better during the earlier stages.

Figure 7.4 is a plot of the comment MRR score against the time difference between the target comment timestamp and the target item stamp. The X-axis represents the difference in time in milliseconds. Yet again, we utilize QB M approximation up till the 99th quantile for a clearer observation of the trend. We aim to observe how the different recommenders perform for older items.
Without a doubt, freshness performs best for fresh items but as time progresses it performs the worst of all recommenders. All other recommenders more or less converge and are on par with the popularity@48 recommender.

**Do comments aid in providing personalized recommendations?**

Considering that on average we were able to retrieve the correct result in the top 5 ranks through the hybrid recommender in an offline scenario, we are confident that adopting the same procedure online would result in personalized recommendations. By accounting for freshness, content and the neighbourhood we were able to exploit all facets of the dataset. Therefore, comments do aid indeed aid in providing personalized recommendations.

### 7.2. Conclusion and Outlook

Comments have evolved over the years, initially it was perceived as a form of providing reactions to web items. Through our work, we have shown that comments when supplemented with the right information will produce personalized recommendations.

More recently, with introduction of various features such as ‘upvote/downvote’, comments have now become more of a struggle to achieve the most votes through witty opinions or providing genuine additive information. The ‘reply’ feature in comments have also turned commenting platforms into conversational mediums leading to heated debates. It is open to question whether these additional meta-information about the comment information improve upon recommendations.

Comments also express explicit sentiment features about the target item, the user would express his hatred or love towards a particular web item. This opens up an entire avenue of questions - Whether to recommend only web items the user has expressed affinity towards in the past? Or does expressing hatred count as expressing interest towards commenting and thereby could also be recommended? We leave these questions open to cater towards potential future work.
A.1. NU.nl Layout

The overall layout of the NU.nl and NUjij.nl interfaces are shown below as perceived by the users.

Figure A.1: Home Page of NU.nl
Figure A.2: Page of Article in NU.nl

Figure A.3: Comments of NU.nl Article in NUjij.nl
A.2. TERMS AND FORMULAE

We will list the different terms and formulae we used throughout the report in the following sections.

A.2.1. TF*IDF WEIGHTS

$t f * i d f$ vectors allow to isolate terms in documents that make them uniquely identifiable from other documents in the corpus. $t f$ stands for term frequency whereas $i d f$ stands for inverse document frequency. In our case, we use the natural logarithm of the $i d f$ as the number of documents in our corpus is large and taking the $i d f$ as is would result in a very large value.

$$t f i d f = f_{t,d} * \log_e \frac{N}{n_t}$$

where,
- $f_{t,d}$ - frequency of term $t$ in document $d$
- $N$ - total documents in corpus
- $n_t$ - number of documents with term $t$

A.2.2. DISTANCE MEASURES

Euclidean and cosine distances are the most common vector space distance measures. Euclidean distance measures the magnitude of the distance between two vectors whereas cosine distance measures the directional difference. In our case, we use cosine similarity which is essentially the inverse of cosine distance and measures the closeness in directional difference.

- **Euclidian Distance**
  $$\text{Euclidian Distance} = ||(q - p)||$$

- **Cosine Similarity**
  $$\text{Cosine Similarity} = \frac{p \cdot q}{||p|| \cdot ||q||}$$

where $||x|| = \sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_n^2}$ is the norm of the vector $x$

A.2.3. MATRIX FACTORIZATION

Matrix factorization is essentially factorization taken to the matrix level, factor matrices are computed through optimization routines, we adopt gradient descent optimization in our case.

For vanilla matrix factorization, the ratings matrix is factorized as,

$$\hat{r}_{u,i} = b_{ui} + q_i^T \cdot p_u$$

where,
- $b_{ui}$ - user-item bias
- $q_i$ - item vector
- $p_u$ - user vector

In our case we handle comments as a form of implicit feedback and SVD++ accounts for implicit feedback situations,

$$\hat{r}_{u,i} = b_{ui} + q_i^T \cdot \left( p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

where,
- $N(u)$ - set of items user $u$ expressed implicit feedback towards
- $y_j$ - implicit feedback vector for item $j$

The parameters in implicit feedback situation are optimized through gradient descent optimization over
20 iterations in the following manner,

\[ p_u = p_u + \gamma \ast (e \ast q_i - \lambda \ast p_u) \]
\[ q_i = q_i + \gamma \ast (e \ast \left( p_u + \left| N(u) \right|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) - \lambda \ast q_i) \]
\[ y_j = y_j + \gamma \ast (e \ast \left| N(u) \right|^{-\frac{1}{2}} \ast q_i - \lambda \ast y_j) \]

where,
- \( \gamma \) - learning rate for that iteration
- \( \lambda \) - normalization scalar

A.2.4. **Weighted Reciprocal Rank Fusion**

Given different recommenders \( R \) weighted reciprocal rank fusion aims to fuse the ranks produced by all \( r \in R \) by computing the sum of the reciprocal ranks produced for each instance. We also prioritize different recommenders over others by offsetting the ranks produced through a variable \( k_r \). The higher the value of \( k_r \) for a particular recommender, the lower the priority it received.

\[ RRF(i) = \sum_{r \in R} \frac{1}{k_r + r(i)} \]

where,
- \( k_r \) - recommender priority
- \( r \in R \) - recommender \( r \) in set of all recommenders \( R \)

A.2.5. **Quantile Bounded Mean**

The quantile bounded mean (QBM) is frequently used throughout many graphs. We utilize this so as to better observe the trend that a graph follows. Consider a sorted series of values \( N_i \) to be a sorted series of values, then the creation of a QBM plot is described below,

- Find 5% quantiles \( N_{q1}, N_{q5}, N_{q10}, ..., N_{q100} \) over the sorted series \( N_i \).
- In some cases we remove the final quantile \( N_{q100} \) as the extremely high values would not be representative of the entire observations.
- Find the average of all values belonging to a particular quantile \( N_{qk} \) and plot the observation.
- Repeat previous step till all quantiles are accounted for.

A.2.6. **Metrics**

Mean Reciprocal Rank (MRR) is simply the reciprocal of ranks produced and averaged over all the observations. This is helpful in discerning the average rank we were able to retrieve a correct prediction.

\[ MRR = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{r_i} \]

where \( r_i \) - rank of instance \( i \)

Precision@1 (P@1) is essentially the precision cut-off till the first rank. We utilize this so as to observe how well our predictors fair in making predictions in the first go. Precision is generally given as,

\[ p = \frac{tp}{tp + fp} \]

where,
- \( tp \) - No. of true positives
- \( fp \) - No. of false positives
In the case of $P@1$, the resulting value is either 1 or 0, we find the average over all such values. For a better understanding of the relevant documents, we also employ the Mean Average Precision@$n$ ($MAP@n$) which is the mean of the average precision@$n$. Average precision@$n$ is slightly different from vanilla precision and is formulated as,

$$AP@n = \sum_{k=1}^{n} (P(k)\Delta r(k))$$

where,

$P(k)$ - Precision@$k$

$\Delta r(n)R$ - change in recall from $k-1$ to $k$

The $MAP@n$ is merely the mean of all $AP@n$. We limit our observations till $MAP@10$ as our assumption is that the user can only perceive 10 recommendations at a time and displaying any more would not create a good user experience.
BIBLIOGRAPHY


