Radboud University

Master thesis computing science

An offline analysis of the CLEF-NEWSREEL evaluation

Author:
Martijn Nanne

Supervisor:
Arjen De Vries

Student number:
s4243692

Second reader:
Martha Larson

July 13, 2017

Abstract

In this research we show an extensive offline analysis of the CLEF-NEWSREEL data 2016. Recommendations have to be made from an emulated stream of data consisting of page views, item updates and click events. Most research is focused on a live setting and use click through rate as evaluation measure.

We show the strengths and weaknesses of different recommendation algorithms on different parts of the data and on different evaluation measures. We show the trade off in the use of session data, the use of page views versus click events and how we handle the exploration/exploitation dilemma.

We set up different experiments where we replay the data in order to test the different recommendation strategies under different circumstances.

We show that the use of other evaluation measures like recall lead to different results. We also showed that click events are very important in news recommendation and should be exploited. We have made the first steps toward hybrid recommenders where we exploit with a most clicked recommender and explore with other recommenders.

Contents

1 Introduction

News websites want to optimize the time spent on their website to increase advertisement revenue. These news websites are continuously looking for new ways to optimize their websites.

One of these ways is to provide news recommendations to the user. These recommendations are provided by an automated system. Conversions by these recommendations could mean that a website could get a competitive advantage by increasing user engagement or time spent on the website.

A news publisher usually publishes a large number of news articles each day. The large amount of articles on the news website causes information overload. A page has only space to provide a couple of recommendations to the user. In order to provide these recommendations we need a recommendation system [18] that will learn which recommendations will be best for the website. The recommendation system has to solve the problem of information overload.

Since 2014, the CLEF-NEWSREEL organization [11] has organized evaluation activities in order to benchmark recommendation systems in the news domain. These evaluations are hosted in collaboration with Plista[1]. Plista is a company that provides recommendation services for online publishers. Whenever a user requests a page, Plista will provide the recommendations that will be shown to the user. In a evaluation, participants have the opportunity to test their recommender systems in an live or offline setting. The offline evaluation consists of a dataset gathered by Plista in the month February 2016. In the live evaluation, participants have to provide recommendations in a live setting to real users on real websites.

1.1 Scope and research questions

In this work we look at the data collected by Plista over the month February 2016. We focus on an offline evaluation of a wide range of recommendation systems. These recommendation systems are evaluated based on the click behavior that was captured in the log data.

The evaluation of CLEF-NEWSREEL is only based on click through rate. This is an important metric, however this metric does not consider the recall of the recommendations. As a result, a system can perform well even if it recommends only a small subset of all possible articles of interest to the user. In this research we compare performance on a set of metrics, to capture more aspects of relevance.

About half of the users in the Plista recommendations are tracked by cookie ids. Therefore, there is a subset of users we have session information for. Prior research has shown that session based recommendations can improve the performance of recommender systems [1]. Yet little research on session based recommenders has been applied to the news domain.

[1] https://www.plista.com/

The Plista data consists of item updates, page views and click events. Earlier research [26] suggested that click events contain a good signal for news recommendations. We would like to see how much difference there is between training recommender systems on page views and on the click events in the dataset. News recommenders have to continuously explore novel new items to recommend. There has to be a trade-off between exploring for new items that are being available on the publishers website and exploiting items that are novel recommendations. In this research we want to explore this trade off.

The main focus of this work will be about the following research questions:

- To what extent can we explain the performance of a recommendation system by means of click through rate?
- To what extent can we exploit session information to improve news recommendations?
- To what extent can we utilize a page view and a click event in news recommendation?
- How can we make the trade off between exploration and exploitation in news recommendation?

1.2 Research challenges

A news website is a dynamic environment, with a continuous stream of new news articles where users usually prefer most recent news articles. Therefore, it is difficult to build up a long term profile of users. Furthermore, many users are not required to log in [6, 17], making it impossible to track the user.

1.3 Novel contributions

This is the first work to evaluate recommendation systems in the CLEF-NEWSREEL evaluation from a variety of angles that reflect different user needs.

We can see that the click behavior of the users brings the most significant signal in terms of click through rate. Most clicked (section 4.1.3) is clearly the best performing algorithm when we evaluate it on click through rate. However if we look at recall, this approach does not perform as well since it only recommends a few different articles. Most popular navigation sequence recommenders (section 4.1.2) [15] seem to be among the best strategies and score fairly well on each evaluation. Furthermore, this research confirms that content based strategies do not seem to perform very well on any of the evaluation measures.

We have made the first steps in using session information in our CLEF-NEWSREEL recommendation systems. We have not been able to get better performance by using session information in order to personalize results.

We showed that recommenders trained only on click event data perform significantly better than recommenders trained on page view data or both page view data and click event data.

We have run experiments using reinforcement learning on the CLEF-NEWSREEL data. We have evaluated the results using only the click events. We have seen that there are significant benefits in hybridizing an explorative recommender with a most clicked recommender, where clicked items are reinforced.

1.4 Practical relevance

This work provides an overview of methods and techniques used in order to provide recommendations in a highly dynamic environment such as news websites. We bring new insights into the CLEF-NEWSREEL evaluation and the field of content recommendation in general. Continued research in this field is relevant because websites want users to be more engaged with the website. Some of the insights in this work can be tested in next year's evaluation.

1.5 Thesis structure

In section 2 we look into previous work related to this work. In section 3 we explain our experimental setup. Here, we explain how the data is structured, detail the preprocessing methods, our evaluation measures and the experiments performed. In section 4, we give an detailed overview of each recommender used in this work. In section 5, we show an analysis of the results. Section 6 provides a discussion about the work and also provide suggestions for future work. In section 7 we answer our research questions based on the results observed in section 4.

2 Previous work

Literature about Recommender Systems has expanded rapidly over the past 20 years. Most of the research is focused on collaborative filtering or a content-based approach. In this section, we discuss previous work on news recommendation, session based recommendation and the exploration/exploitation dilemma in the news domain.

2.1 News Recommendation

We start to provide an overview of previous work in the field of news recommendation, with a focus on work related to the CLEF-NEWSREEL evaluation.

Recommender systems are used to reduce information overload [18]. One news portal will usually publish many articles a day. However, the "screen real estate" is limited; we cannot promote all the articles that could be of interest. In CLEF-NEWSREEL that is build on a real life platform for news recommendation, each page request, we can only recommend between 1 and 6 recommendations. Therefore it will become hard to select which articles to show to a user.

News articles usually become less relevant to the users as they get older. In order to give a recommendation of an item based on a collaborative filtering algorithm, there have to be interactions with an item first. However, fresh articles do not have these interactions yet. This is called the *cold start problem* in recommender systems [19]. Liu et al. [14] reported that the Google News system needed several hours in order to collect enough clicks in order to recommend new news stories to their users. A half day latency is a problem in the news domain as it is dependent on delivering news in a timely manner.

Recommendations should not only be recent, they should also be relevant to the users. There is a recency-relevancy trade off when recommending news articles [4]. The lifespan of a news item is dependent on the publishers domain and the popularity of the article [15]. For example, a small news article has a relatively short lifespan compared to an extensive background article. On the Newsreel domains KSTA and Motor-Talk we can see a significant drop in the average amount of impressions after 12 hours of publishing the article in Figure 1. The rank in Figure 1 is the popularity of the articles falling in that category. For example, rank 1..250 is the average over the 250 most popular items .

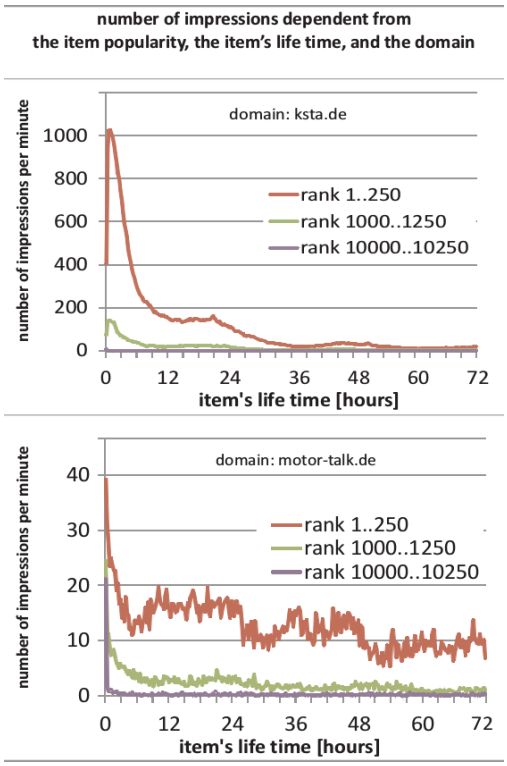

Figure 1: Impressions over time

2.2 CLEF-NEWSREEL

The CLEF-NEWSREEL evaluation is comprised of two tasks to evaluate news recommender algorithms [11]. For researchers, there has been little opportunity to evaluate recommender systems in a live environment. It is important to evaluate algorithms in a live environment because industry tends to evaluate algorithms in a live setting. CLEF-NEWSREEL provides the opportunity to evaluate your algorithms both in a live and an offline setting. CLEF-NEWSREEL closes the gap between evaluation strategies found in industry and academia.

It is organized as an evaluation for researchers to test their news recommender systems in order to make significant advances in this field. This work will not focus on the live environment but will try to replicate the live setting using the data provided for offline evaluation. Earlier research has also used this approach to evaluate their algorithms in a "*near to online*" setting [15, 4]. In this setting, the data will be replayed in the same way as the live setting, implying a data stream of events. These events consists of page views, click events and item updates. A streaming or online setting puts additional constraints on recommender systems. Factorization methods are often used in a

batch processing manner, making it difficult to apply to streams [15]. Furthermore, the stream is dynamic in nature, meaning that items have a short time to live and user behavior changes over time [6]. The behavior of users is different on each publisher's website [15]. For example, users read articles in different time windows, read a wider range of articles or are more likely to read popular articles.

Personalized news recommendation in the Google News domain builds up a long term user profile but also uses recent local news trends [14]. In the CLEF-NEWSREEL evaluation, participants cannot construct a long term data profile about users. Users often do not have to log on in order to read the news. Many users are not tracked at all and user profiles can be noisy because they are only partially tracked [17]. CLEF-NEWSREEL provides us with only one month of log data to evaluate our recommender algorithms, making it difficult to build up a long term profile of interests of a user.

Another difference between CLEF-NEWSREEL and the aggregated news websites like Google News is that Plista does have data about different publishers but it cannot recommend items from one publisher on another publishers website. The users are also not tracked across different domains.

2.2.1 Overview of CLEF-NEWSREEL Recommenders

In this section we provide an overview of the recommenders used in the CLEF-NEWSREEL evaluations:

Popularity based recommenders - These recommenders are based on the popularity of an item [15, 6, 7, 12]. The time window of considered recommendations has a significant effect on the accuracy of the recommendations. Recommendations based on the most popular items of the last week perform worse than if we base our recommendations on the most popular items of the current day [15].

Another recommender strategy that is based on popularity is the most clicked recommender strategy. The difference with the most popular recommender is that this recommender bases popularity only on click events. The CLEF-NEWSREEL evaluation broadcasts click events to all participants. The most clicked algorithm was the winning recommendation strategy from 3-9 April 2016 [26]. The number of clicks on items seem to follow a power law, where the top six most clicked items continuously make up for more than 80% of the total amount of clicks. However, the set of items that make up the majority of clicks is dynamic and the time duration for when an article is a top six most clicked items is not the same for each item [26].

The most popular based recommender does not take any user context into account.

Content based recommenders - These recommenders are based on the similarity between the item being read and other recent items.

Different approaches are mentioned in literature. CWI has used the title and the preamble in order to make content based recommendations. Apache

Lucene was used in order to determine similar articles. Other participants used the conventional TF-IDF vector space model on the title and summary [6].

Content based approaches have had varying success in the CLEF-NEWSREEL evaluation [17, 6, 15, 26]. Content based approaches do not take popularity into account and often fail to pick up on the most interesting news stories. Content based approaches redirects users to similar contents. Little research has been done on combining popularity with content similarity in the CLEF-NEWSREEL evaluation.

Category based recommenders - A category based recommender is based on the similarity between the category of other items and the item currently being viewed [17]. The category of an item are present in the page view or click event context. A popularity based approach has also been performed on categories. Items that intersect with the category of the currently viewed items category are considered within a popularity based recommender [6]. Moderate success has been achieved using this approach.

Most popular navigation sequence - This approach is based on the transition of viewed items [15]. For each article, a statistic is kept of which article is requested next by users.

The most popular navigation sequence recommender aims to take an important context variable into account. The context variable is the page which a user is currently viewing. The most popular sequence recommender aims to better adapt to the particular context than the most popular recommender.

Lommatzsch et al. [15] have been the only team testing this approach. The most popular sequence was the best performing recommender on a couple domains in their research. These results are based on data from Plista in 2014.

User based collaborative filtering - User based collaborative filtering will aim to recommend similar news items to similar users. Users will give implicit feedback about their preferences through item views. The similarity between users is directly computed on the interactions [15].

A problem with this approach is that this approach is still suffering from the cold start problem.

Geolocation based recommender - Based on the idea that people from similar locations prefer similar articles [8, 6, 14]. However, this approach has not led to significant results so far.

2.3 Session based recommendations

Other fields of study have explored the use of session information in order to make predictions. Session based recommenders use the users current session information in order to make recommendations [22].

In the context of search, Bennett et al. [2] have studied the use of short term (session) behavior in combination with long term (historic) behavior in order to make personalized predictions. Their research showed that the use of session information provided a major gain in extended sessions.

Session based recommenders have also been an active field of study in the domain of item recommendation in e-commerce. At Recsys 2015, a challenge

was organized to address the problem of item recommendation in e-commerce using session information [1]. There are many situations where the users of e-commerce websites are not identified. However, users are still tracked on the website. In this context, using session information has been shown to help improve performance when making recommendations [25, 9].

Session information can also be leveraged as a post filtering mechanism [12]. For example, a most popular recommender can be post filtered by the interests showed inside a session. This can be done based on feature matching, cooccurrence scores or recently viewed.

Yet little research has been done in the field of session based recommendation services in the news recommendation domain. To our knowledge, none of the participants attempted to leverage session information in the CLEF-NEWSREEL evaluation. We could only argue that the most popular navigation sequence recommender takes session information into account.

2.4 Exploitation vs Exploration

Traditional approaches such as Collaborative Filtering seem to fail to take novelty into consideration. Many recommenders are too greedy in their recommendations, predicting the items with the highest predicted ratings or highest click through rate [24, 23]. The recommended items are not new and unexpected and the user may not be aware about other content that is available on the website. When intent is unclear or uncertain, discovery should always be the goal [20].

Reinforcement learning is a different strategy to make recommendations. It uses feedback from the user in order to reinforce desired actions. One key element in reinforcement learning is exploration. It means that rather than always taking the most optimal action for to get the short term benefit, we take a suboptimal action to gather information. This information can lead to an even better performance in the future [23]. When a recommender is too exploitative, it might not discover items that might maximize the amount of clicks. A fraction of all user actions could be used in order to do such exploration. However, if we use too many actions to gather this information, we might hurt the short term recommendations too much. This trade off is known as the exploration/exploitation dilemma [10, 23].

News items are only relevant for a relatively short period of time. Therefore it is necessary to discover when items are relevant and when items become irrelevant. There is a classic exploitation versus exploration problem in news recommendation. We constantly want to exploit the items that do well in our recommendation strategy while also exploring new items that will become the most relevant items in the future.

There is no research on this topic within the CLEF-NEWSREEL use case to our knowledge.

3 Experimental setup

3.1 Data & Data Preprocessing

The data used in this work is a log dump by Plista [13]. The log dump was performed during Febuary 2016. The dataset consists of 58000 item updates, 2 million click event notifications and 168 million page views. The data can be replayed as a stream of events. Each item in the dataset has a specific timestamp so we are sure that the events are replayed in the same order as the live event.

The replayed events consist of three types:

- Page View
- Click event
- Item Update

Contextual information has been provided for each page view. A page view consists of a timestamp, the user id of the user, the id of the viewed item, the publisher id and contextual information. The user id of the page view belongs to a specific publisher, and is therefore unique to one publisher. We are not able to track a user over different domains.

The data provides a variety of contextual fields including category, keywords, age group of the user and geographical location. For a full explanation of the contextual fields we refer to the dataset website[2]. Notice that most records contain only a subset of all possible contextual fields.

Click events have the full information about the page from where a recommedation has been clicked and provides the information about which recommendation has been clicked.

Item updates contain the `created` timestamp and the `updated` timestamp. The `created` timestamp contains the point in time when the article was initially created. If the article is then updated, the `updated` timestamp will show the timestamp of the update. Item updates also contain the title of the article and a short preview about the article. Finally, there is a url link to the published article.

We want to mimic the real world situation where we get events into the system in the same way as it would be in a live setting. Therefore we stream the data in the temporal order of the log data.

Table 1 shows the number of clicks per publisher. Out of the eight domains occurring in the dataset, only seven have click events. We can see that three more domains (publisher ids 596, 3336 and 13556) contain an insufficient number of clicks to form the basis of a statistically meaningful evaluation. We ignore these 4 sets in our evaluation in the rest of this work. We do not know the publisher names of the ids mentioned without a name because they do not occur in any item update.

[2] http://www.clef-newsreel.org/dataset/

Publisher	Clicks
Sport1	1498076
Tagesspiegel	403211
KSTA	194165
Gulli	13813
3336	45
13554	2
596	1

Table 1: Total amount of clicks per publisher

Table 2 provides an overview of the unique item updates and unique items clicked. Most domains have more unique items clicked than there are item updates, the only publisher that has more unique items clicked than updates is the publisher Tagesspiegel. This implies that at the point of recommendation, the content of many of the clicked items is not known, because the item updated corresponding to creation of the item on the website is not part of the data dump.

Publisher	unique item updates	unique items clicked
Sport1	1669	2272
Tagesspiegel	5423	2719
KSTA	1073	2086
Gulli	23	168
3336	0	19
13554	0	1
596	0	1

Table 2: Total amount of clicks per publisher

Table 3 shows the number of views made by users with a cookie id versus those by users without a cookie id. We can see that the number of clicks and the number of views (and their ratio) differ per publisher. Note that a large fraction of the total number of page views results from users known by their cookie id.

Publisher	views unknown users	views known users
Sport1	59.125.428	58.609.907
Tagesspiegel	4.089.966	17.213.414
13554	2.503.951	6.657.483
KSTA	2.122.357	9.472.782
Gulli	784.261	2.086.350
3336	1.607	6.526
2522	40	18
15739	1	10

Table 3: Total amount of clicks per publisher

We have created a subset of the data where we only account for users that contain a session of more than two page views in order to evaluate the click through rate on this part of the data. We only use this subset of the data when we evaluate our session based recommenders which we discuss later in this manuscript. We count one day as one session, considering all page views of a user on that day to belong to the same session. When we discuss a session in this manuscript, we use this definition of a session.

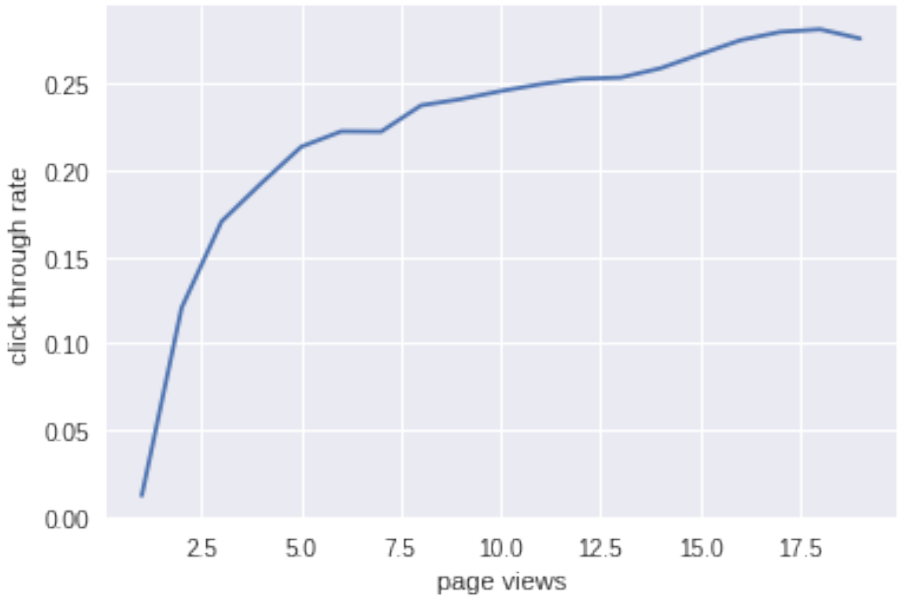

Figure 2: CTR per session length page views

In Figure 2 we can see the click through rate for a session with x number of page views. We see an increasing trend which suggests that there is a higher click through rate when pages are requested later in a session. We can see a quick rise in click through rate at 1 - 5 views within a session. The datapoints in the graph are based on at least 100 users in the dataset a day.

3.2 Evaluation

In our evaluation we use the *near-to-online* evaluation methodology [15]. In *near-to-online* evaluation, we replay the Plista dataset as if it were a live setting, which means that we stream all samples in the Plista dataset. In the live setting, participants are asked to provide recommendations to each recommendation request. In our case, we use the click events for our evaluation.

The number of recommendations to be made varies between one and six in the live competition. Whenever we receive a click event, we do not know how

many recommendations had to be given, therefore we chose to always provide six recommendations.

The information that is available to the recommender is derived from all views, events and item updates that have occurred before this point in time. No future information from the data set will be used in order to make these recommendations. If the clicked item occurs in these six recommendations, it is counted as correct, otherwise it is counted as not correct. We only use the click events in our evaluation.

A short summary of our evaluation process:

1. For every click event received
2. Extract the page viewed and the item clicked from the click event
3. Make 6 recommendations based on this page and its contextual information
4. Evaluate if the clicked item occurs in our recommendations

We apply two restrictions to considered recommended items:

- The recommended items have not been seen by the user
- Items without an identifier will not be recommended

3.2.1 Click through rate

The Newsreel competition uses click through rate as the final evaluation measure [11]. The click through rate is calculated in the following way:

$$CTR = \frac{correct\_recommendations}{total\_nr\_recommendations}$$

A correct recommendation is added whenever one of the six recommendations is correctly predicting the clicked item. We only use click events for our evaluation. Total recommendations means the total number of recommendations made.

We calculate the CTR for each publisher on each day of the data stream. An average CTR is calculated over all days (the average of the CTR of each day).

3.2.2 Recall

Aside from a CTR based evaluation, we wanted to consider a measure that rewards recall:

$$Recall = \frac{unique\_items\_correct}{total\_nr\_unique\_items\_clicked}$$

Each click event is handled as described above in our evaluation process. However, instead of calculating the click through rate, we calculate the recall.

We keep track of all unique items that are predicted correctly and we keep track of all unique items clicked in the data stream. Each day, we compute the recall as described above. An average recall is computed over all days (the average of the recall of each day).

3.2.3 Cumulative Gain

If an article is just very popular, it might be less interesting to the user. The user might not get the best experience from the service by just seeing the same items all the time. The user might want to be "surprised" sometimes.

We defined a third measure with the objective to favor less obvious recommendations and thus favor a diverse set of recommendations though not ignoring the popular articles. We have developed this measure at the end of our research and aim to make the first step in order to define a measure that captures both CTR and recall in news recommendations. Each clicked recommendation will have a relevancy score which will be added to the total gain score. The relevancy score will be added to the gain score whenever we make the recommendation. The relevancy score will decay as there are more clicks on the same item.

$$Rel(item) = \frac{1}{\sqrt{times\_clicked(item)}}$$

$$CG = \sum_{item}^{correct\_items} Rel(item)$$

Again, we only use the clicked items in this evaluation and calculate the cumulative gain over each day and than calculate the average gain over the gain of each day. We keep track of the times an item is clicked over all days. We do not reset the times an item is clicked in the evaluation at the start of each day.

3.3 Experiments performed

In this section I motivate the setup of four different experiments, each related to one of the following questions:

A How do the different recommenders, behave under different evaluation measures?

B What is the right trade off between session and non-session based recommenders?

C Should recommenders be trained on views or on events?

D How should the trade off between exploration and exploitation be made?

3.3.1 Recommender Overview (A)

In these experiments, we will use all data available to test the recommendation algorithms mentioned in the next section. We follow the approach of [6] to flush all statistics any recommender has each day, because it had the highest performance in this research compared to other time windows. Statistics will be kept separately for each recommender and each publisher.

We are going to compare the different types of recommendation systems on each of the evaluation measures. The result is the performance over time for each recommender and an averaged performance over all domains.

3.3.2 Session based Recommenders (B)

In this part, we only use the session data. We only make predictions on click events where we have session information, where we have at least two views by this user on the same day. We will compare different algorithms where we have a baseline algorithm which will only look at the currently viewed item and a modified algorithm that takes the whole session into account.

We will compare the session based recommender with its corresponding recommender based only on the currently viewed item.

3.4 Views vs Clicks (C)

In this part, we pick a couple of recommender systems and train each of those recommender systems on seperate parts of the data. We train each recommender on the page view data, the click event data and on both page views and click events. We compare the performance of these recommenders on each part of the data.

3.4.1 Exploitation vs Exploration (D)

The CLEF-NEWSREEL evaluation broadcasts all click events to all participants. Therefore the participants can utilize click events on recommendations made by other recommender systems. The system a participant creates should take into account that it has to reproduce the competition setting to achieve the same result in a live setting. For example, a most clicked algorithm in the competition should take into account that it needs the other algorithms in the competition in order to discover click events. However, the evaluation in the competition only evaluates on the participants part of the recommendations regardless of the click events used by other recommeders.

In a real system we do not have the click events from other recommenders unless we combine the most clicked recommender with other recommenders. We need a recommendation engine to be both explorative and exploitive. We cannot use a most clicked algorithm without determining which articles are clicked at all. In these experiments we want to make the first steps toward hybrid recommenders that take exploitation and exploration into account. We are going

to make a combination of multiple recommenders that can explore for novel new items while exploiting items that prove to be successful recommendations.

In these experiments, the recommenders will only use click event information if the recommender has actually made a correct prediction. We therefore emulate a situation where we have no other information than the information we get from our own recommender system.

4 Method

This section introduces the different recommender strategies used in our research. We provide an explanation along with pseudocode for each recommender strategy. First, we will assess a couple of basic recommender strategies that will be evaluated on all data. Next, we will show a couple of recommender strategies that incorporate session information. Finally we will show the hybrid recommenders that will make the exploration/exploitation trade off and will only use correctly predicted click events.

4.1 Recommender Overview

4.1.1 Popularity based recommender [15, 6, 7, 12]

A popularity based recommender provides recommendations based on the popularity of items on a specific domain. The recommender recommends the most viewed pages at a specific time.

For each item, we count the number of times the item is accessed. Counts are stored in a dictionary. For each publisher, we keep track of the counts in a separate dictionary. Whenever a recommendation is requested, we make a lookup in the dictionary of the specific publisher and order the counts in descending order. The resulting recommendations are the top 6 recommendations requested, these are the most viewed items of the specific publisher at that point in time.

We use both the page views as well as the click events in order to calculate this statistic. A page view is not added to the item if a user has already requested the item before.

Every hour, we remove the statistics that are not in the 250 most popular items on a domain for peformance reasons.

```
Data: Recs and Events
while Input do
    read nextInput;
    if nextInput = recommendation_request then
        add_view(publisher, viewed_item_id);
    else if nextInput = Event_Click then
        make_recommendations(publisher);
        add_view(publisher, from_page);
        add_view(publisher, clicked_page);
    end
end
```

Algorithm 1: Popularity based recommender

4.1.2 Most popular sequence recommender [15]

This recommender learns which item is requested most after each article. We learn this statistic from both the page views as well as the click events. When

a user has clicked a recommendation, we know that these pages are clicked in sequence. We also count sequences of different page views by the same user. We are not able to use page views by an unknown users.

Also, we do not put any time constraint on this factor. If a user has observed page X at 10AM in the morning and visited page Y in the evening, we still count (X,Y) as a sequence.

```
Data: Recs and Events
while Input do
    read nextInput;
    if nextInput = recommendation_request then
        if already_seen(nextInput.user) then
            add_view(item_id, prev_seen_id);
            user.prev_seen_id := item_id;
        else
            user.prev_seen_id := item_id;
        end
    else if nextInput = Event_Click then
        make_recommendations(item_id);
        add_view(item_id, rec_id);
        add_view(item_id, prev_seen_id);
        user.prev_seen_id := rec_id;
    end
end
```

Algorithm 2: Most popular sequence based recommender

4.1.3 Most clicked

The most clicked recommender recommends the articles that are most clicked thus far. This recommender counts the clicked recommendations.

```
Data: Events
while Input do
    read nextInput;
    if nextInput = Event_Click then
        make_recommendations(item_id);
        add_click(rec_id);
    end
end
```

Algorithm 3: Most clicked recommender

4.1.4 Cooccurrence based recommender

This recommender is based on the cooccurrence of clicks on news articles by the same user. For each combination of articles, we compute the amount of cooccurrences between articles. More clearly, if user A views articles X - Y - Z, and user B views articles Z - Y, we get the cooccurrence scores (X, Z, 1), (X,

Y, 1), (Y, Z, 2). Whenever user C views article Z, it recommended Y because it has the highest coocurrence score with Z.

We also include the event data in the statistics. Clicks from unknown users are counted as a cooccurrence between the page from where this recommendation was clicked and the clicked item.

Whenever we have to make a recommendation, we look up the item id in the dictionary and pick the top 6 items that have cooccurred the most with the currently viewed page.

```
Data: Events and Updates
while Input do
    read nextInput;
    if nextInput = recommendation_request then
        store_combinations(item_id, user_id, user.seen_items)
        add_seen_item(user.seen_items, item_id)
    else if nextInput = event_click then
        make_recommendations(item_id) store_combinations(item_id,
        user_id, user.seen_items) store_combinations(clicked_item, user_id,
        user.seen_items)
    end
end
```

Algorithm 4: Cooccurrence based recommender

4.1.5 Stem overlap based recommender

A content based recommendation system recommends content similar to content that a user has liked in the past [16]. This recommender makes recommendations based on the similarity between the currently viewed item and other items. We do this by computing the stem overlap [15] over the title and text of the article and other items published by the same publisher. We apply stemming and stop word removal to improve performance.

The items considered are the items that have occurred in the item updates up until the current point in time.

```
Data: Events and Updates
while Input do
    read nextInput;
    if nextInput = item_update then
        remove_stop_words(nextInput);
        stem_words(nextInput);
        store_stemming(nextInput);
    else if nextInput = event_click then
        overlap_dict := compute_overlap(nextInput.item, publisher_items);
        sort_on_overlap(overlap_dict);
        make_reccommendations(overlap_dict);
    end
end
```

Algorithm 5: Stem overlap based recommender

4.1.6 Keyword based recommender

Each item contains a specific set of keywords. This recommender will try to find similar items based on these keywords. We store the keywords for each article in a dictionary. Keywords are provided in the context of a page view and a click event. Whenever we need to provide recommendations, the system will find the most similar articles based on the keyword overlap.

```
Data: Events and Updates
while Input do
    read nextInput;
    if nextInput = view then
        store_keywords(item_id)
    else if nextInput = event_click then
        keywords := get_keywords(item_id);
        overlap_dict := compute_overlap(publisher, keywords);
        sort_on_overlap(overlap_dict);
        make_reccommendations(overlap_dict);
        store_keywords(item_id)
    end
end
```

Algorithm 6: Keywords based recommender

4.1.7 Most popular topic

We could believe that there are specific news topics that are very popular. This recommender looks at the most popular item sequences and then tries to find articles similar to these items. In order to do this, we use the keyword recommender to find the most similar articles around the most popular item. We recommend the four most popular items together with two most similar

items.

```
Data: Events and Updates
while Input do
    read nextInput;
    if nextInput = view then
        store_keywords(item_id);
        store_view(item_id);
    else if nextInput == event_click then
        keywords := get_keywords(item_id);
        popular_items := get_poprecs(item_id);
        sim_items := get_similar_items(popular_items);
        recommendations := popular_items + sim_items;
        make_recommedations(recommendations)
    end
end
```

Algorithm 7: Most popular topic based recommender

4.2 Session recommenders

If we only use the users currently viewed page, there may be little personalization. Many of the incoming recommendation requests are from a known user. In these cases, we might be able to exploit past impressions in order to make a personalized recommendation.

In this case, we do not only look at what the most popular sequence will be from the currently viewed page, but the algorithm will also account for the items that were previously seen in the session.

Older page views have less value in order to predict recommendation for the current point in time. Therefore we add a weight decay for older articles. The weight decay is initially set to 0.5 determined by a few explorative experiments. We also performed extensive experiments using different weight decay measures using the most popular sequence session recommender.

4.2.1 Most popular sequence session recommender

In the session based most popular sequence recommender, we look at each item in a user session and determine the most popular sequences for each of these items. These counts are added up and result in the final recommendations.

```
Data: Recs and Events
parameter: weight_decay
while Input do
    read nextInput;
    if nextInput = recommendation_request then
        if already_seen(nextInput.user) then
            add_view(item_id, prev_seen_id);
            user.prev_seen_id := item_id;
        else
            user.prev_seen_id := item_id;
        end
    else if nextInput = Event_Click then
        foreach item in seen_items_today do
            total_mpc_count += mpc_count(item) * weight_decay;
        end
        make_recommendations(total_mpc_count);
        add_view(item_id, rec_id);
        add_view(item_id, prev_seen_id);
        user.prev_seen_id := rec_id;
    end
end
```

Algorithm 8: Most popular sequence session based recommender

4.2.2 Cooccurrence session based recommender

Like the session based mpc recommender, we look at every item the user has already viewed. We count all cooccurrences from the viewed items. We recommend the items with the most cooccurrences with the items the user has already viewed.

```
Data: Events and Updates
parameter: weight_decay
while Input do
    read nextInput;
    if nextInput = recommendation_request then
        store_combinations(item_id, user_id, user.seen_items);
        add_seen_item(user.seen_items, item_id);
    else if nextInput = event_click then
        foreach item in seen_items_today do
            total_cooc_count += cooc_count(item) * weight_decay;
        end
        make_recommendations(item_id);
        store_combinations(item_id, user_id, user.seen_items);
    end
end
```

Algorithm 9: Cooccurrence session based recommender

4.3 Exploration/Exploitation recommenders

In this section we will present a set of hybrid recommenders [3]. These are combinations of the recommenders mentioned in section 4.1. The goal of these recommender algorithms is to be both explorative and exploitive in behavior. We want to achieve an algorithm that can discover which items are going to be most clicked and then exploit the most clicked algorithm. At the same time, we still want to explore to identify new items that could potentially be a new most clicked item.

We use an epsilon greedy based strategy [23] in order to explore new options. Every recommendation made by the most clicked algorithm has a chance to recommend from the explorative recommender instead. We have used $\epsilon = 0.2$ in all of our experiments. Although 0.1 would be a typical value for epsilon [21], we wanted to be more explorative as the news domain is dynamic in nature and new articles have to be constantly discovered. However, an extensive search over this has to be performed to determine the best value for this parameter. We did not have sufficient time to do this and therefore used 0.2.

The explorative recommender has to be based on views rather than click events.

4.3.1 Popularity based recommender and most clicked

This recommender is explorative by using the most popular based recommender. The popularity based recommender tries to find popular items from the views. Once these popular items are also clicked a lot, the most clicked recommender

will reinforce itself and will start to exploit these items.

```
Data: Events and Updates
parameter: ϵ
while Input do
    read nextInput;
    if nextInput = recommendation_request then
        pop_rec.store_view(nextInput)
    else if nextInput = event_click then
        if train_phase then
            succes := pop_rec.make_recommendations;
        else
            recs := most_clicked_rec.make_recommendations();
            foreach rec in recs do
                if rand(0, 1) > ϵ then
                    rec := pop_rec.get_rec();
            end
            succes := make_recommendations(recs);
        end
        if succes then
            pop_rec.store_click();
            most_clicked_rec.store_click();
    end
end
```

Algorithm 10: Popularity based recommender and most clicked hybrid

4.3.2 Keyword based ranker and most clicked

The keyword based recommender does not need any click or view count information in order to find similar items. It will try to explore by recommending

similar items as the user is currently viewing.

```
Data: Events and Updates
parameter: ϵ
while Input do
    read nextInput;
    if nextInput = recommendation_request then
        pop_rec.store_view(nextInput)
    else if nextInput = event_click then
        if train_phase then
            succes := keyword_rec.make_recommendations;
        else
            recs := most_clicked_rec.make_recommendations();
            foreach rec in recs do
                if rand(0, 1) > ϵ then
                    rec := keyword_rec.get_rec();
            end
            succes = make_recommendations(recs);
        end
        if succes then
            keyword_rec.store_click();
            most_clicked_rec.store_click();
    end
end
```

Algorithm 11: Keyword based recommender and most clicked hybrid

4.3.3 Most popular sequence and most clicked

We have also made an hybrid recommender where we explore with most popular sequences over the views.

```
Data: Events and Updates
parameter: ϵ
while Input do
    read nextInput;
    if nextInput = recommendation_request then
        mpc_rec.store_view(nextInput)
    else if nextInput = event_click then
        if train_phase then
            succes := mpc_rec.make_recommendations;
        else
            recs := most_clicked_rec.make_recommendations();
            foreach rec in recs do
                if rand(0, 1) > ϵ then
                    rec := mpc_rec.get_rec();
            end
            succes := make_recommendations(recs);
        end
        if succes then
            mpc_rec.store_click();
            most_clicked_rec.store_click();
    end
end
```

Algorithm 12: Most popular sequence and most clicked hybrid

5 Results & Analysis

This section discusses the results from our experiments, in order of description of the experimental setup in section 3.3. Table 4 describes the mapping of the algorithms mentioned in the results.

Name	Full Name	Data used
poprec	Popularity based recommender	page views + click events
coocrec	Cooccurence based recommender	page views + click events
most clicked	Most clicked recommender	click events
mpseq	Most popular sequence recommender	page views + click events
mpseq clicks	Most popular sequence recommender on click events	click events
keywordrec	Keyword overlap based recommender	page views + click events
stemrec	Stem overlap based recommender	item updates
most popular topic	Most popular topic recommender	page views + click events
session mpseq	mpseq session based recommender	page views + click events
session mpseq clicks	mpseq clicks session based recommender	click events
session coocrec	coocrec session based recommender	page views + click events
mpseq views	mpseq trained only on page view data	page views
poprec views	poprec trained only on page view data	page views

Table 4: Mapping of result terminology

5.1 Recommender Overview (A)

5.1.1 CTR

In this section we describe the performance in terms of click through rate. Table 5 displays the average performance of each recommender on every domain.

Recommender	Average CTR
coocrec	0.263
poprec	0.164
most clicked	**0.764**
mpseq	0.348
stemrec	0.007
mpseq clicks	0.624
keywordrec	0.017
most popular topic	0.496

Table 5: Average CTR over all domains

We can see that the recommenders that are only based on click events outperform the other recommenders. Most clicked is definitely the best performing recommender in terms of click through rate. The content based recommenders have the lowest click through rate.

Figure 3: Gulli

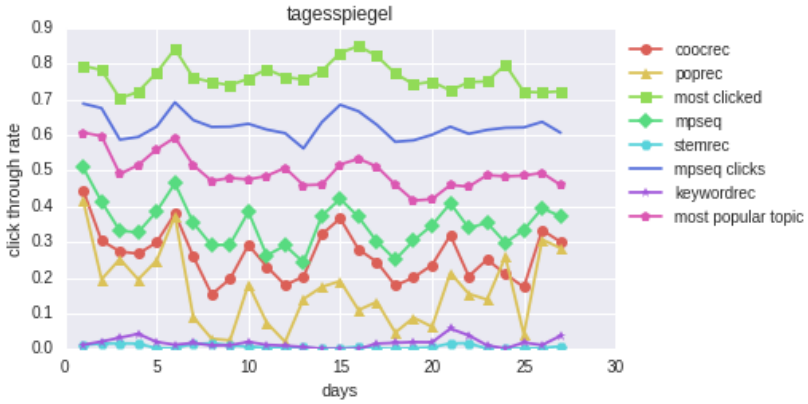

Figure 4: Tagesspiegel

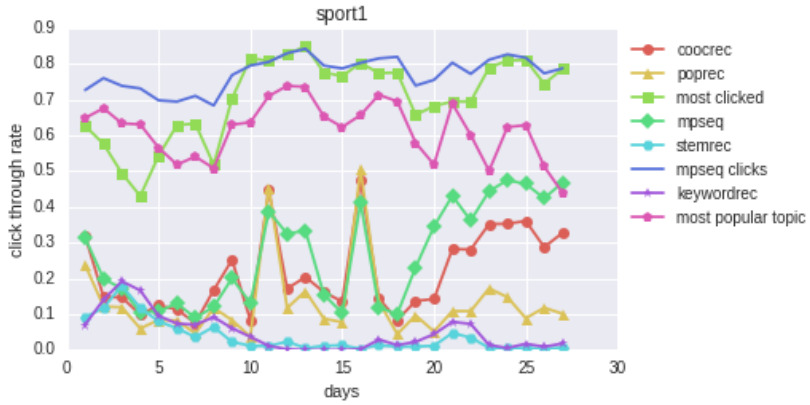

Figure 5: Sport1

Figure 6: KSTA

We can see that the predictive power of the most clicked recommender is just slightly different over different domains. We see stronger performance differences in other recommenders. For example, we see that the performance of the most popular sequence recommender is higher in the Tagesspiegel domain than the Gulli domain. Stem overlap based recommendations do perform much worse.

5.1.2 Recall

We can see that the performance in terms of recall, which is the number of uniquely correct predicted items, is quite different from the CTR performance.

While in table 5 the most clicked recommender performs best, in Table 6 we see that the popularity based recommenders are not as performant.

Recommender	Average Recall
coocrec	0.287
poprec	0.105
most clicked	0.137
mpseq	0.305
stemrec	0.073
mpseq clicks	**0.345**
keywordrec	0.196
most popular topic	0.276

Table 6: Average Recall over all domains

The highest recall score results from the most popular sequence recommender trained only on the click data. This means that we obtain a higher recall score from the recommendations when we make recommendations using a most popular sequence recommender than if we make recommendations using a most popular recommender.

The stem overlap based recommender has the worst performance, on recall as well as CTR.

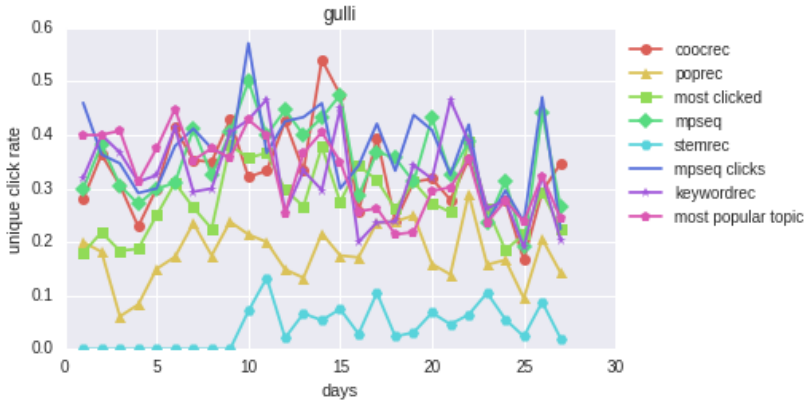

Figure 7: Gulli

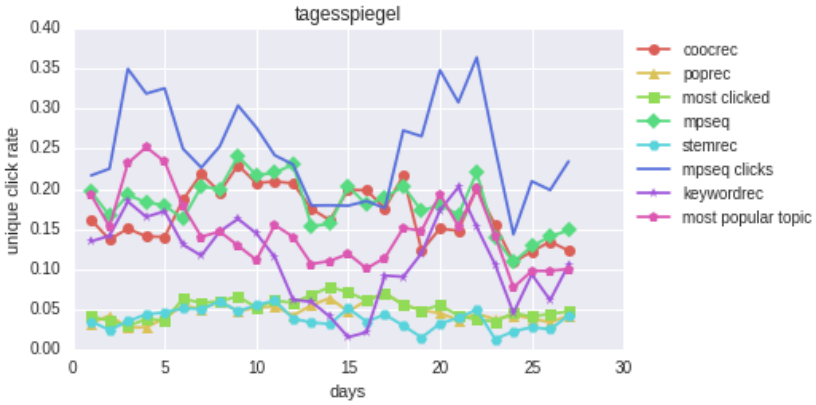

Figure 8: Tagesspiegel

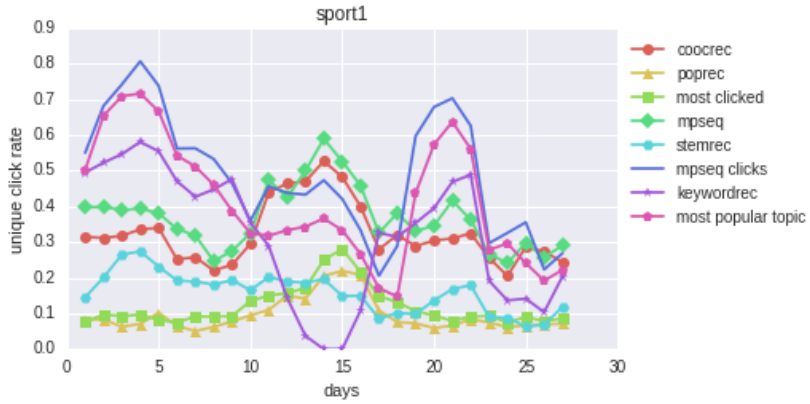

Figure 9: Sport1

Figure 10: KSTA

The performance of the most popular sequence recommender trained on clicks is not always the best performing recommender in terms of recall. For example, we can see that in Figure 10 the performance of coocrec and most popular sequence trained on all data are outperforming most popular sequence trained on click events on the last 15 days of the competition.

5.1.3 Gain

The gain measure is used to see which recommender gets a considerable amount of clicks on a wide range of articles. We can see that on this measure, the most popular sequence trained on clicks has the best performance. It performs considerably better than the most popular clicked recommender.

Recommender	Average Gain
coocrec	198
poprec	132
most clicked	229
mpseq	203
stemrec	27
mpseq clicks	**323**
keywordrec	50
most popular topic	230

Table 7: Average gain over all domains

Most clicked has a worse performance on this measure. The most clicked algorithm only recommends the top 6 most clicked items, causing the gain of these items to become small for each click. However, since the click through

rate is much higher, it still performs similar to other recommenders like the cooccurrence based recommender.

5.2 Session based Recommenders (B)

In this section, we will go over the effect of using session information in our recommendations. Note that we used a subset of the data where we only made predictions on click events where the user had at least 2 page views. In order to get these results, we only used the click events where we could use session information in order to make the recommendations.

Recommender	Average CTR
mpseq clicks	0.500
session mpseq clicks (decay 0.5)	0.526
coocrec	0.238
session coocrec (decay 0.5)	0.267
mpseq	0.242
session mpseq (decay 0.5)	0.289

Table 8: Average CTR over all domains

When we compare each recommender with its corresponding session based recommender, we can see that all session based recommenders performed better when we look at CTR.

Recommender	Average Recall
mpseq clicks	0.343
session mpseq clicks (decay 0.5)	0.304
coocrec	0.298
session coocrec (decay 0.5)	0.255
mpseq	0.317
session mpseq (decay 0.5)	0.291

Table 9: Average Recall over all domains

When we look at Table 9, we can see that in contrast to the CTR performance, the recall performance is worse when we use session based recommenders. The amount of different recommendations is lower.

Recommender	Average CTR
mpseq clicks session (decay 1)	0.520
mpseq clicks session (decay 0.8)	0.524
mpseq clicks session (decay 0.6)	0.527
mpseq clicks session (decay 0.5)	0.528
mpseq clicks session (decay 0.3)	0.529
mpseq clicks session (decay 0.2)	0.530
mpseq clicks session (decay 0.1)	0.531
mpseq clicks session (decay 0.0)	0.500

Table 10: Average CTR over all domains

In Table 10 we show the performance of most popular sequence session based recommender trained on click data with different decay values. There is a performance improvement when we go from a 100% decay (using only the currently viewed item) to weight decay 0.1. This is a very large decay, therefore the performance improvement is arguably the cause of using session information. However, when we use the whole session using the same weight (decay = 1) the performance is still better than if we did not use any session information.

Recommender	Average Recall
mpseq clicks session (decay 1)	0.280
mpseq clicks session (decay 0.8)	0.287
mpseq clicks session (decay 0.6)	0.298
mpseq clicks session (decay 0.5)	0.304
mpseq clicks session (decay 0.3)	0.320
mpseq clicks session (decay 0.2)	0.328
mpseq clicks session (decay 0.1)	0.335
mpseq clicks session (decay 0.0)	0.340

Table 11: Average Recall over all domains

Table 11 shows that if we add more weight to session information, we see a lower recall score. In our experimental setup, we have not been able to get a higher recall score by incorporating session information.

5.3 Views vs Clicks (C)

Recommender	Average CTR
mpseq views	0.071
mpseq clicks	0.575
mpseq	0.307
poprec views	0.150
most clicked	0.752
poprec	0.203

Table 12: Average CTR over all domains

In table 12 we can see that the performance of the same recommender trained only on click data is much better than if we train the recommender only on the view data. If we train the recommenders on both views and clicks, we get a score in between the view based score and the click based score. We can conclude that if we use both clicks and views, it is beneficial to train only on clicks and leave out the view data.

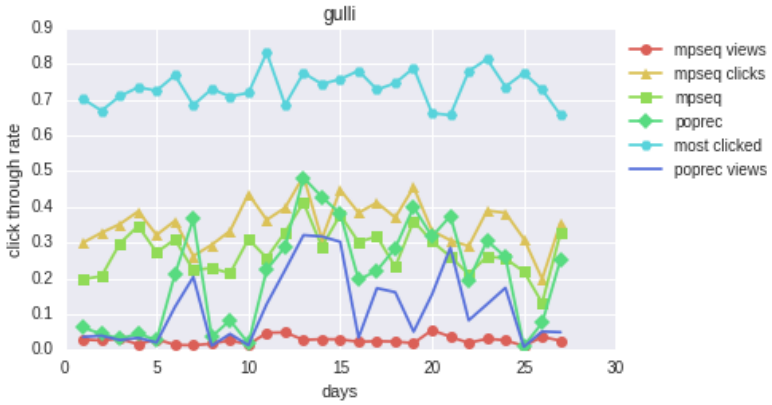

Figure 11: Gulli

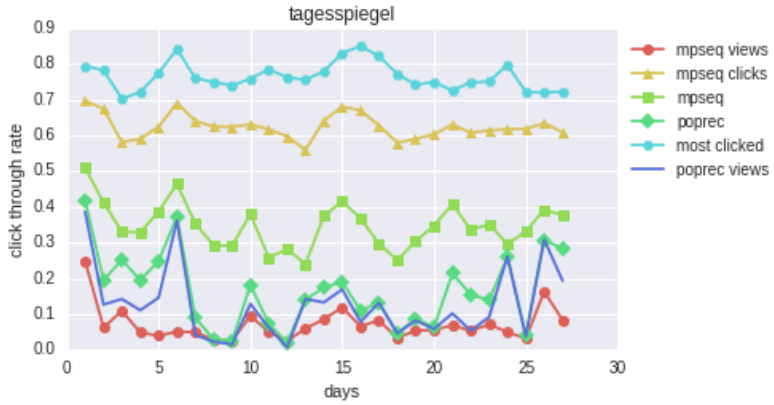

Figure 12: Tagesspiegel

Figure 13: Sport1

Figure 14: KSTA

We can see that the trade off between training on clicks or views is consistent over all domains. However, the benefit of using only click events does depend on the domain.

Recommender	Average Recall
mpseq views	0.237
mpseq clicks	0.342
mpseq	0.302
poprec	0.103
most clicked	0.135
poprec views	0.098

Table 13: Average Recall over all domains

Table 13 shows that not only the CTR is better if we train on click data, also the recall is higher. Training on views clearly worsens the performance on both CTR and Recall.

5.4 Exploitation vs Exploration (D)

The performance of the recommenders are clearly worse due to the lower amount of training data than when we do not use all click data. We can see that the performance of the hybrid between most clicked and the most popular sequence recommender has the best overall score while using click through rate as evaluation measure. We can see a clear improvement in performance in each of the hybrid recommender variants, when compared to their corresponding single recommender.

Recommender	Average CTR
poprec	0.164
mpseq	0.243
most clicked + poprec	0.289
most clicked + keywordrec	0.229
most clicked + mpseq	**0.541**

Table 14: CTR over all domains (for reference, most clicked has a CTR of 0.764)

We see that most popular sequence recommender trained on all data has a CTR of 0.348 (see Table 5). However, in the emulated setting where we can only use correctly predicted click events, the CTR is 0.243. The difference in performance is the result of "free" exploration done by other recommender systems in the CLEF-NEWSREEL evaluation since these are click events that are generated by other recommenders.

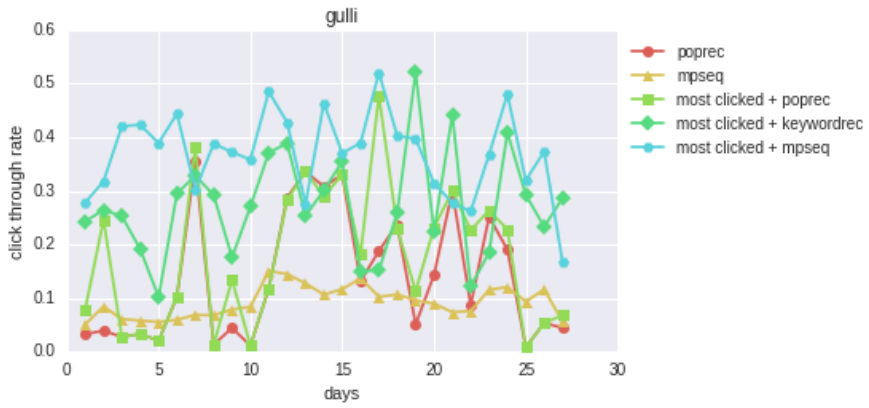

Figure 15: Gulli

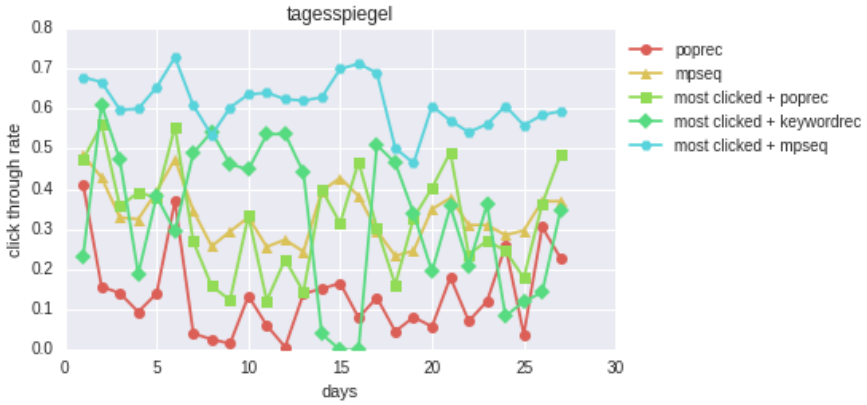

Figure 16: Tagesspiegel

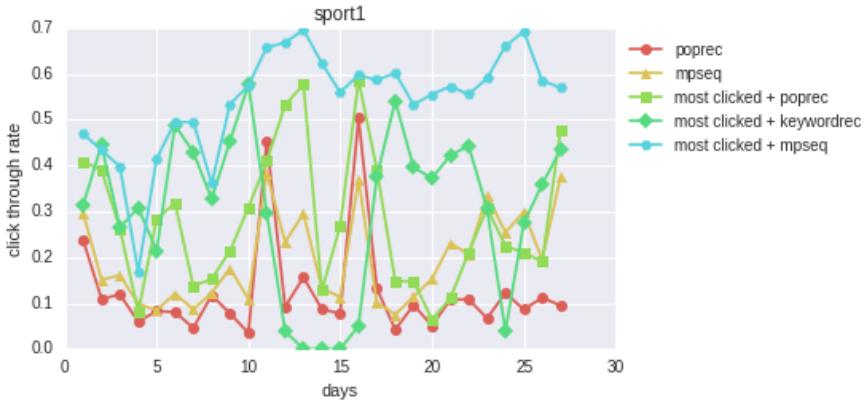

Figure 17: Sport1

Figure 18: KSTA

In the above figures, we can conclude that most popular sequence recommender is the best explorative recommender on all domains. The performance of all recommenders is worse on the Gulli domain.

In Table 15, we show that on recall the most popular sequence recommender still has the best performance. When we look at the hybrid recommenders, the most popular sequence - most clicked hybrid has the best performance.

Recommender	Average Recall
poprec	0.097
mpseq only	**0.289**
most clicked + poprec	0.090
most clicked + keywordrec	0.130
most clicked + mpseq	0.185

Table 15: Recall over all domains

6 Discussion

We want to start off the discussion by talking about using the click through rate as the main strategy in order to evaluate recommender systems in the news domain. We have seen in our results that if we use the broadcasted click events, we can get the highest click through rate by recommending the most clicked items. Other strategies, that include the data from views perform significantly worse. In practice, we cannot use only click event data, since a recommender system always has to find out which articles are being clicked on. In the competition setting, participants get "free" exploration where they can use click events from recommendations made by other participants.

When we look at the recall and cumulative gain, we do get different results from the recommender systems. Most popular sequence has a higher average recall score and a higher gain score. On some domains we can see that at certain time windows, other recommenders had a higher recall score. For example coocrec and most popular sequence trained on all data outperformed most popular sequence trained on click events in this time window. Reason for this could be that the group of most popular sequences in the click event data was very small while correct sequences or cooccurrences could still be learned from the page views.

Since the click events have such a high bias in terms of results in click through rate, it would be interesting to see how participants would score on either a different measure like gain or when they can only use the click events from their own recommendations. In a live setting, the system does only get feedback from its own recommendations. However, in the competition researchers might use click events explored by other participants. We have shown in the results that when we train a most popular sequence recommender on all data we get a better CTR and Recall than if we can only use the click events that are correctly predicted. The difference in performance is what we call the effect of "free exploration" when we use all data. If we want to replicate a live setting as much as possible and give researchers a chance to compare their results with other recommender systems, we have to make this evaluation.

Most clicked algorithms have a very low recall score. The recommendations will always be the same for every user and no personal recommedations are given. It might give a user a better experience to be surprised by the recommendations. A trade off has to be made between sustaining a sufficient click through rate and giving unexpected and interesting recommendations.

Content based approaches did not perform well in our experiments. The content based approaches were not very sophisticated. Since the most clicked items consist of only a couple of items that make up for a large portion of the clicks, the content based approaches are unable to capture these characteristics. Content based approaches that do not take popularity into account will therefore not be successful. However, our most popular topic recommender also did not perform better than most popular sequence.

Relatively simple approaches seem to perform really well. Classic approaches as collaborative filtering or factorization methods are hard to execute due to the

dynamic nature of the data. Focus should be made towards approaches that can quickly discover novel new items to recommend.

Using session information seems to be an interesting field in news recommendation. In our data analysis we have seen that longer sessions have a higher click through rate. However, in our experiments we have not been able to successfully personalize recommendations using session information. We see that regardless of our decay parameter, we do get a higher CTR when we use session information. However, it is arguable that the improvement is caused by the experimental setup. The reason for this is that sometimes the recommender does not have sufficient statistics to make recommendations based on the currently viewed page. However if this is the case, it might have sufficient statistics for pages earlier in the session and is therefore able to make recommendations based on these items where it couldn't make these recommendations without the session information.

The use of session information in our experiments lowered recall, the recall drops even more when we increase weight decay. We argue that the use of session information in our experiments caused the recommendations to recommend more popular items. When a few items are very popular it is not uncommon that personalization tends to recommend the most popular items. In [5] the goal is to give personalized recommendations for tourism. However, a couple of places are very popular and recommender systems tend to recommend the most popular items when personalizing results. The same problem seems to occur in the news domain where a couple of news articles are extremely popular.

When we take into account that we can only use feedback from our correctly predicted click events we observe a benefit in the use of hybrid recommender systems. One part of this hybrid has to be the most clicked recommender. By using a simple $\epsilon - greedy$ algorithm we can improve our click through rate performance.

We have to take into account that all results are based on an offline evaluation, derived from a set of recommender algorithms running at that time. There will be a bias in the way the data is gathered. If we successfully model the recommenders used at the time, we have a higher chance to get a better performance. However, since the data was gathered in a period where participants tested different recommender algorithms, we do feel confident about our results.

7 Conclusion

We started off trying to answer the following research questions:

- To what extent can we explain the performance of a recommendation system by means of click through rate?
- To what extent can we exploit session information to improve recommendations?
- To what extent can we utilize a page view and a click event in news recommendation?
- How can we make the trade off between exploration and exploitation in news recommendation?

We have seen a difference in peformance of our recommenders on different evaluation measures. Previous work has stated recall to be an important factor while making recommendations. Since click through rate does not consider any recall measure we recommend to also evaluate on a similar measure. However, we can only use recall in the offline evaluation.

We have not been able to personalize results by using session information. However, we have seen some positive results in CTR by using the session information. Further research has to be done to personalize recommendations.

Recommenders trained on click events clearly outperform recommenders trained on page views when we look at click through rate. To increase click through rate performance, click information has to be utilized in order to increase performance.

The last question came from the fact that in a live setting, we do not know which items will be clicked. Participants should be aware of the "free exploration" of clicked items by other participants in the evaluation. A live recommender can be exploited by reinforcing most clicked items. We have shown different exploration strategies and shown we're able to get benefits by combining one exploration strategy with the most clicked recommender.

8 Future work

How users would respond to the recommendations in a live setting will remain a question until we test these algorithms in a live setting. In the next evaluation window, it would be interesting to see if we can test some of the approaches we have described in this research in a live setting since we were not able to do so this year due to the timing of the competition.

In this research we have used the session information by adding up the statistics from each viewed item by a user. We might be able to get further improvements by taking into account which page views in the sessions are more unique and give more weight to these items. Also, we did not normalize the scores on each page viewed. For example, if the most popular sequence recommender did not get many transitions from a certain page, the scores we'll add up do not add much weight to the final recommendations. Further research could provide smarter ways to utilize session information.

In order to personalize results, we should overcome the problem where personalization methods stimulate the recommendation of popular items. Further research has to be done in order to overcome this problem.

It would be interesting to see if we can successfully deploy a hybrid recommender in a live setting where we show that a hybrid between a most clicked recommender and the most popular sequence recommender outperforms the most popular sequence recommender.

Further improvements can be made to the hybrid recommenders that perform both exploration and exploitation. It would be interesting to see which hybrids make the best trade off between exploration and exploitation.

Content based recommenders seemed to perform very weak in this research. However, it would still be interesting to see if content similarity measures could be used in order to make recommendations. For example, we could use cross domain content similarity measures to determine popular topics. We did not use any cross domain knowledge to improve recommendations and we intend to perform further research in this field.

9 Code

The code can be found on Github [3].

[3] https://github.com/martijnnanne/newsreel-offline

References

[1] David Ben-Shimon, Alexander Tsikinovsky, Michael Friedmann, Bracha Shapira, Lior Rokach, and Johannes Hoerle. Recsys challenge 2015 and the yoochoose dataset. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 357–358, New York, NY, USA, 2015. ACM.

[2] Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 185–194, New York, NY, USA, 2012. ACM.

[3] Robin Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

[4] Abhijnan Chakraborty, Saptarshi Ghosh, Niloy Ganguly, and Krishna P Gummadi. Optimizing the recency-relevancy trade-off in online news recommendations. In *Proceedings of the 26th International Conference on World Wide Web*, pages 837–846. International World Wide Web Conferences Steering Committee, 2017.

[5] Maarten Clements, Pavel Serdyukov, Arjen P. de Vries, and Marcel J.T. Reinders. Using flickr geotags to predict user travel behaviour. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 851–852, New York, NY, USA, 2010. ACM.

[6] Doychin Doychev, Aonghus Lawlor, Rachael Rafter, and Barry Smyth. An analysis of recommender algorithms for online news. In *CLEF (Working Notes)*, pages 825–836. Citeseer, 2014.

[7] Doychin Doychev, Rachael Rafter, Aonghus Lawlor, and Barry Smyth. News recommenders: Real-time, real-life experiences. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 337–342. Springer, 2015.

[8] Gebrekirstos G Gebremeskel and Arjen P de Vries. The degree of randomness in a live recommender systems evaluation. In *CLEF (Working Notes)*, 2015.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[10] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, 16(1):63–90, 2013.

[11] Frank Hopfgartner, Torben Brodt, Jonas Seiler, Benjamin Kille, Andreas Lommatzsch, Martha Larson, Roberto Turrin, and András Serény. Benchmarking news recommendations: The CLEF-NEWSREEL use case. In *ACM SIGIR Forum*, volume 49, pages 129–136. ACM, 2016.

[12] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. Adaptation and evaluation of recommendations for short-term shopping goals. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 211–218. ACM, 2015.

[13] Benjamin Kille, Frank Hopfgartner, Torben Brodt, and Tobias Heintz. The plista dataset. In *Proceedings of the 2013 International News Recommender Systems Workshop and Challenge*, NRS '13, pages 16–23, New York, NY, USA, 2013. ACM.

[14] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.

[15] Andreas Lommatzsch and Sahin Albayrak. Real-time recommendations for user-item streams. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1039–1046. ACM, 2015.

[16] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[17] Alan Said, A Bellogın, and Arjen De Vries. News recommendation in the wild: Cwi's recommendation algorithms in the CLEF-NEWSREEL challenge. In *Proceedings of the 2013 International News Recommender Systems Workshop and Challenge. NRS*, volume 13, 2013.

[18] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.

[19] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[20] Brent Smith and Greg Linden. Two decades of recommender systems at amazon.com. *IEEE Internet Computing*, 21(3):12–18, 2017.

[21] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[22] Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 17–22. ACM, 2016.

[23] Stephan Ten Hagen, Maarten Van Someren, Vera Hollink, et al. Exploration/exploitation in adaptive recommender systems. *proceedings of Eunite 2003*, 2003.

[24] Zhe Xing, Xinxi Wang, and Ye Wang. Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. In *ISMIR*, pages 445–450, 2014.

[25] Peng Yan, Xiaocong Zhou, and Yitao Duan. E-commerce item recommendation based on field-aware factorization machine. In *Proceedings of the 2015 International ACM Recommender Systems Challenge*, RecSys '15 Challenge, pages 2:1–2:4, New York, NY, USA, 2015. ACM.

[26] Jing Yuan, Andreas Lommatzsch, and Benjamin Kille. Clicks pattern analysis for online news recommendation systems.