Research Repository UCD

Title	News Recommenders: Real-Time, Real-Life Experiences
Authors(s)	Doychev, Doychin, Rafter, Rachael, Lawlor, Aonghus, Smyth, Barry
Publication date	2015-06-03
Publication information	Doychev, Doychin, Rachael Rafter, Aonghus Lawlor, and Barry Smyth. "News Recommenders: Real-Time, Real-Life Experiences." Springer, 2015.
Conference details	UMAP 2015: 23rd International Conference on User Modeling, Adaptation, and Personalization, Dublin, Ireland, 29 June - 03 July 2017
Publisher	Springer
Item record/more information	http://hdl.handle.net/10197/8442
Publisher's statement	The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-20267-9_28 .
Publisher's version (DOI)	10.1007/978-3-319-20267-9_28

Downloaded 2024-04-18 04:47:46

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information

News Recommenders: Real-time, Real-life Experiences

Doychin Doychev, Rachael Rafter, Aonghus Lawlor, and Barry Smyth

Insight Centre for Data Analytics,
University College Dublin Belfield, Dublin 4, Ireland.
{doychin.doychev,rachael.rafter,aonghus.lawlor,barry.smyth}@
insight-centre.org
http://www.insight-centre.org/

Table of Contents

Cover Page 1
D. Doychev et al.
1 Introduction 2
2 Related Work 3
3 Plista 4
4 Challenges 5
5 System Architecture 6
6 Algorithms 7
6.1 Popularity-based Algorithms 7
6.2 Similarity-based Algorithms 8
7 Setup and Methodology 8
7.1 Experimental Setup 8
7.2 Results and Analysis 10
7.3 Lessons Learned 10
8 Conclusions 10

Abstract. *In this paper we share our experiences of working with a real-time news recommendation framework with real-world user and news data. We discuss the challenges faced while working in such a noisy but uniquely real-world context. Specifically, we focus on an initial evaluation of a 12 different news recommendation algorithms across 7 different German news sites, including general news, sports, business, and technology related news sites. We compare the performance of these algorithms, paying particular attention to their relative click-through rates and how this can vary with time of day and news domain.*

Keywords: News Recommender Systems, Real-time Recommendation Frameworks, Adapting/Contextualizing Recommendations

1 Introduction

Recommender systems have become an essential part of our day-to-day lives, when it comes to dealing with the overwhelming amount of information available, especially online. Recommender systems improve user experience and increase revenue in the context of online retail stores (Amazon, eBay), online news providers (Google News, BBC) and many more. In this work we present a wide range of online recommender algorithms and compare their performance in the scope of the CLEF-NewsREEL 2014 online challenge - *has to be modified to fit the current paper*. In CLEF-NewsREEL 2014, participating teams connect to the Plista Open Recommendation Platform [1, 2] and have to respond to real-time user requests for recommendations. Given that recommender systems have

traditionally been evaluated offline, this poses an interesting challenge in terms of algorithm efficiency, tracking users, building quality models of user browsing behaviours and preferences, and in terms of dealing with a highly dynamic domain like news in which there is a constant cycle of new articles appearing and older articles becomming redundant. We consider the challenges of online news recommenda- tion more fully in Section 3.1. The rest of this article is organised as follows: in Section 2 we present some related research in news recommendation; in Section 3 we provide a more detailed description of the Plista ORP framework[3] and the CLEF-NewsREEL challenge, the challenges of online news recommendation, and our system architecture; in Section 4 we describe our recommender algorithms; in Section 5 we report and analyse the data collected and the performance of our recommender algorithms, and in Section 6 we conclude and discuss directions for future research.

2 Related Work

Within the field of recommender systems, the problem of recommending news articles to readers has a number of unique and interesting features. In many traditional application domains of recommender systems a user profile is often available, for example: movies that have been rated or products purchased. The user must create a profile which is used to build a ratings or preference history which is associated with that user as they interact with the site and these detailed profiles then feed into the creation of personalised recommendations. In the news domain it is generally not common to have detailed user profiles, with users not often required to sign in or create profiles. It is also not common for users to rate news articles and often the only information available is implicit in the logs of the users click patterns. This presents a particular challenge for collaborative filtering methods which rely on the opinions of similar users to generate recommendations [5], [9], [7].

Further complications for collaborative filtering arise from the dynamic nature of the users and the news items themselves [6],[3]. In general, users will prefer fresher news articles, and building an accurate model of user preferences based on the items they have previously read can be difficult [4], [6]. While users may have preferred categories of news articles, or topics they are particularly interested in, these preferences are difficult to learn. User preferences change over time too, and another challenge is to provide a diverse set of interesting recommendations, accounting for known users preferences and recency and popularity of the news articles themselves [8].

Content based approaches can run into problems where some measures of similarity identify news articles which are in fact about different topics. Extracting the constantly changing distribution of topics in the news presents a challenge [12] in addition to learning how users choices are influenced by these latent factors [11].

3 Plista

Plista provides a platform ORP [1] for making live recommendations to a number of their client sites. Plista communicates via an HTTP API by sending (JSON) messages. One can register a team name and as many algorithms as one like. However, we should note that Plista sees each algorithm individually, meaning that all the different types of messages described below are replicated and if the team uses a single machine this can introduce an unnessesary overhead in terms of network traffic. In Section 7.1 we will see how we overcome this challenge.

The event messages, Plista sends us, are triggered by a user reading an article or clicking on a recommendation (*event notification*), and whenever an article is created or updated by the publisher (*item update*). Requests for article recommendations (*recommendation request*) are sent as message. As an indicator of whether a recommender framework performs as expected (responding to the above 3 types of messages within a certain timeframe - 100ms; responding with a correctly formated recommendation message; and whether the machine used slows down when under load or the network bandwidth is big enough) Plista sends *error notifications*. Even though Plista is fault tollerant towards error notifications, they are particularly important for monitoring the general health condition of the recommendation framework as exceeding a certain threshold per unit time results in a temporary ban and concequently in loss of data.

The event notification information that we receive is somewhat incomplete. Specifically, teams are told which items were recommended and which item was clicked by the user. However, when a click event occurs we can only guess the set of recommended items it was part of. This missing data could aid us in our analysis. The resulting dataset is unique in many respects, providing detailed user profile information for some users (where available), cross-site data (from 13 - do we keep it as what the paper sais or what is currently available 11 (accodring to event notifications), or 8 according to item_update or 7 as the ones we consider? different news providers), and information about different interaction types. The dataset is fully described in [10].

With each message type (request, event, update) the Plista ORP framework provides additional metadata regarding the current user and article. Although the Plista ORP API documentation [1] lists almost 60 fields of information, in practice we found many were unclear, or not useful or detailed enough to use. Fields such as *Pixel_4th_party* and *Filter_allowosr* typify the ones we simply didn't understand (they had no description either), and popular demographic signals like age, gender and income, expressed as probabilities (male vs female, age and income brackets) turned out to be too vague to be of use in reality. In the end, we settled on 10 fields for use in our recommenders:

geo_user The geolocation of the user
time_weekday The day of the week
category The subcategory under which an article is published within a given domain, e.g. sport, world, local etc.

[1] orp.plista.com

time_hour The hour of the day
item_source The unique item (article) identifier
publisher The publisher (domain) of an article.
keyword Keywords with their corresponding counts occurring in the article
user_cookie The user id (recall users are not obliged to register so many do not have a unique id.
title The title of the news article
summary The summary of the news article

Note that the *category*, *publisher* and *keyword* fields only provide a numerical id rather than a textual representation. For *category* and *publisher* it was possible to exploit URLs in order to determine this information, but for keywords there was no way to uncover what the words actually were, or how they were chosen. Nonetheless, we found that despite this the keywords still provided a useful signal. *don't know about this sentence yet* In Section **??** we provide a deeper analysis of the data provided and used.

4 Challenges

In this section we present the challenges concerning the production of online news recommendations. In the section that follows, we detail the system architecture that we have implemented to cope with these challenges and we describe our recommendation algorithms, suitable for this environment.

Traditionally, recommender systems are evaluated offline, with plenty of time to build complex models of user browsing behaviours and preferences. In real-time online scenarios like NewsREEL however, such leisure is not afforded; not only do teams have to respond within a very tight time frame (100ms), they also have to deal with factors like the lack of rich user browsing history as users are not obliged to register and therefore do not always have persistent profiles or identifiers. Moreover, a user can access the news sites from different devices, and many users can do so from the same shared device, further complicating the ability to reliably track their browsing, as pointed in [10]. Tracking their preferences is also non-trivial, as users do not provide any explicit feedback on recommendation quality, and clickstream data is the only signal of relevance available. Finally, the nature of the news domain itself throws its own set of challenges into the mix due to the dynamic nature of the data, where many new article appear every hour, and older articles quickly become redundant. In such unstructured and dynamic environments, it is necessary to apply techniques that satisfy requirements such as response time, scalability while improving the user experience using limited and sometimes noisy information. Adomavicius et al. [2] point out that these sorts of environments pose serious challenges to standard collaborative filtering recommenders. Instead we should use other recommendation algorithms that fit the nature of the data and take advantage of the features provided by Plista for each user, and the various properties of the articles themselves [1].

5 System Architecture

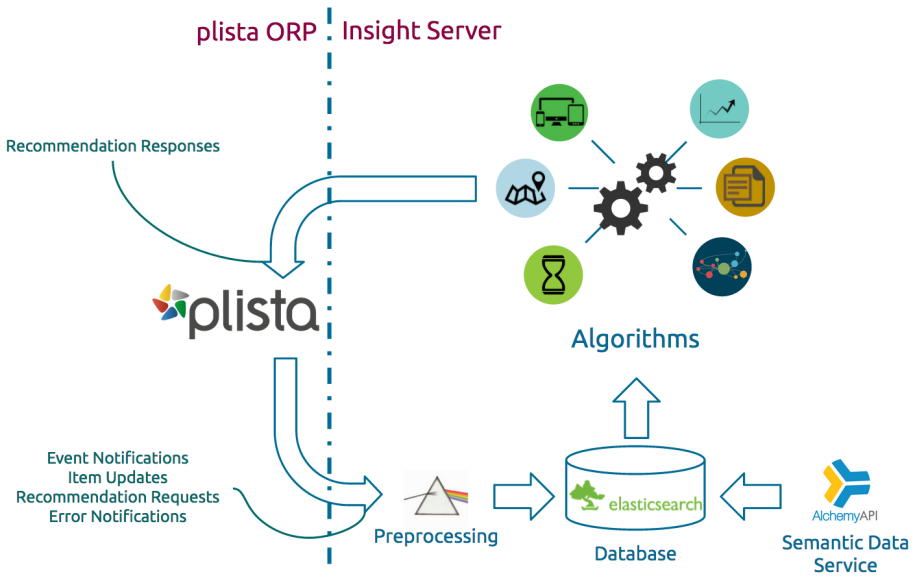

Fig. 1. System Architecture

Our system architecture, shown on Fig 1, is implemented in Python and is designed to accomplish our goals of scalability, speed and extensibility. The first point of interaction with Plista are four web servers, which implement the standard Python web protocols, each responsible for the different types of Plista requests (event notifications, item updates, recommendations and error notifications). All servers write to a database, for which we use ElasticSearch[2].

Rather than trying to guarantee accurate recommendations in under 100ms, we precompute the recommendations and store them in ElasticSearch. For each of our recommendation algorithms, we have a long running process which continually reads the latest events and articles from the database to build its recommendations. With this offline approach, there is a danger that we might send back a recommendation which contains outdated, non-recommendable articles, during the time it took to compute the recommendations. To minimise this possibility, we update our recommendations in the background as frequently as possible so that our recommendations have a minimal lag. For most of our algorithms, the typical refresh time is less than 5 minutes. We have constructed the system with a goal of compromising between accuracy or freshness and scalability.

[2] http://elasticsearch.org/

6 Algorithms

In this section we describe our recommender algorithms. For each we describe the algorithm by how it selects the *candidate set* i.e. the larger set of items that will be considered for recommendations, and its *ranking* strategy, i.e. how it ranks the candidate items for recommendation, before returning the *top N* to the user. We refer to the *target user* as the user for whom the recommendations are required, and the *target article* as the news article the user is currently reading - the recommendations will appear on this page.

We have implemented 12 recommenders in total: 6 popularity-based recommender and 6 content-based recommenders.

Before each algorithm is run, we apply three basic filters to the item set:

Exclude items from other domains : Articles must come from the same domain as the current article. Note that although we get user and news article data from multiple web sites it is prohibited to make cross-site recommendation, contradicting what the authors of [10] say. However, the Plista dataset let us observe user behaviours across multiple web sites and as part of our future work we plan to use them to better understand and target individual users.

Exclude already seen articles : Clearly we do not make recommendations of articles we know the target user has already seen.

Exclude non-recommendable items : These are items that are flagged as non-recommendable (usually older articles) by Plista.

6.1 Popularity-based Algorithms

Each of the 6 popularity recommenders rank their candidate set by item popularity, i.e. the number of users who have read an article. However, they differ in the way they compile their candidate sets. We have developed a basic as well as more advanced recommenders that use additional features. They can be described, based on whether and what information they use as a filter, as follows:

Popularity - No Filter The candidate set is all items in the dataset.

Popularity - Geolocation The candidate set is all items that have been read by users in the same geographical location as the target user.

Popularity - News Category Every item is associated with 0 or 1 news category (business, sports, politics, etc.). The candidate set is all items whose category intersects with the target article's category.

Popularity - Day of Week The candidate set is all items that have been seen at least once in the same day of the week as the target article.

Popularity - Hour of the Day The candidate set is all items that have been seen at least once in the time range [current hour - 1, current hour + 1], where current hour is the hour of the day in which we receive the recommendation request.

Popularity - Positive Implicit Feedback The candidate set is all articles that have been successfully recommended in the past to some user, i.e. clicked by the user. The more popular an article is as a recommendation, i.e. the more clicks it has, the higher the algorithm ranks it.

6.2 Similarity-based Algorithms

This group of similarity-based recommenders find articles that are similar to the current article for recommendation to the target user. The intention here is not to present the user with an article essentially the same as the current one, something which would of course be undesirable, but rather to find articles that are strongly related to the current article. Given our recommender algorithms only operate within a given domain, the chances of the recommendation being too similar to the current article are low.

All the similarity-based recommenders we deploy use the conventional TF-IDF vector space model to derive the candidate set. However, each algorithm uses a different selection of content within an article over which to apply the model. For each of the following similarity-based recommenders, the candidate set is always ranked in decreasing order of similarity to the current article.

Similarity - Title The articles are represented by the terms in their title.

Similarity - Summary The articles are represented by the terms in their summaries only.

Similarity - Title and Summary The articles are represented by the terms in their titles and summaries only.

Similarity - Full Body Text The articles are represented by the terms in their titles and full body text. As Plista provides only the title and the summary of the articles, to apply this algorithm we have a long running process which craws through the news articles' web pages and extracts the full body text using boilerpipe[3]

Similarity - Keywords : The terms are the keywords provided by Plista (recall that Plista provides a set of keywords and their frequencies within a document, and although they are publisher difined we do not know what they are as in our dataset they are represented numerically).

Similarity - German Entities Using AlchemyAPI[4] we extract entities, in German, from the full text of the articles as their representation.

7 Setup and Methodology

7.1 Experimental Setup

Check List

1. time periods - done

[3] https://code.google.com/p/boilerpipe/
[4] http://www.alchemyapi.com

2. table of the domains and their description - done
3. volumes of data - done
4. round robin, reasons behind it - done
5. data we consider, volumes, in order to calculate CTR - done
6. how do we calculate CTR - done

We ran our experiments from 21/12/14 to 31/12/14, inclusive, giving us a total of 11 day worth of data. During this period we had access to 7 news websites, listed with their description and unique identifiers (ID) in the table 1. During this period we gathered some 14.2M user logs (event notifications), over 3.1M recommendation requests of 2.48M unique users and some 20k unique news articles.

Table 1. News web site details with their Plista suppied unique ID (Domain ID), name (Domain Name) and brief description.

Domain ID	Domain Name	Description
418	ksta.de	Regional news website
1677	tagesspiegel.de	National news website
596	sport1.de	National sports website
2522	computerwoche.de	Technology & Business website
3336	tecchannel.de	Computer Hardware & Software news website
694	gulli.com	IT & Technology website

In order to let our framework be extensible with a minimal footprint on performance, both in terms of system load and network traffic, we register a single algorithm (remember in Section 3 we said that we can register multiple algorithms to a single team) and then we use a Round-robin approch to rotate our 12 algorithms giving them equal oppurtunity of generating recommendations.

Plista provides us with the visual mean of observing CTR performance of registered algorithms. However, we work with a real-life dataset and we want to calculate our own CTR, which imposes some constraints on the subset of data we look at. We follow the following step when we filter our dataset:

- Consider only successful recommendations, filtering out recommendation requests that have missing data needed for the algorithm to execute or requesting for data that is not present in our database.
- Consider recommendation request which have both item and user present and they are not represented by default values.
- Consider recommendations, notifications, and updates only from 7 domains

After this processing step we are left with some 900k recommendations over which we will conduct hour CTR analysis. We calculate CTR with the following formula

$$CTR = \frac{clicks}{impressions} \times 100 \tag{1}$$

where clicks is the number of recommendations where users clicked on 1 news article out of set of 6 and impressions is the total number of recommendations.

7.2 Results and Analysis

Check List

1. recommendations rate
2. clicks compared to recommendations
3. clicks compared to recommendations on per domain basis barchart
4. General CTR
5. CTR per hour
6. CTR per domain
7. clicks compared to recommendations on per domain basis as points

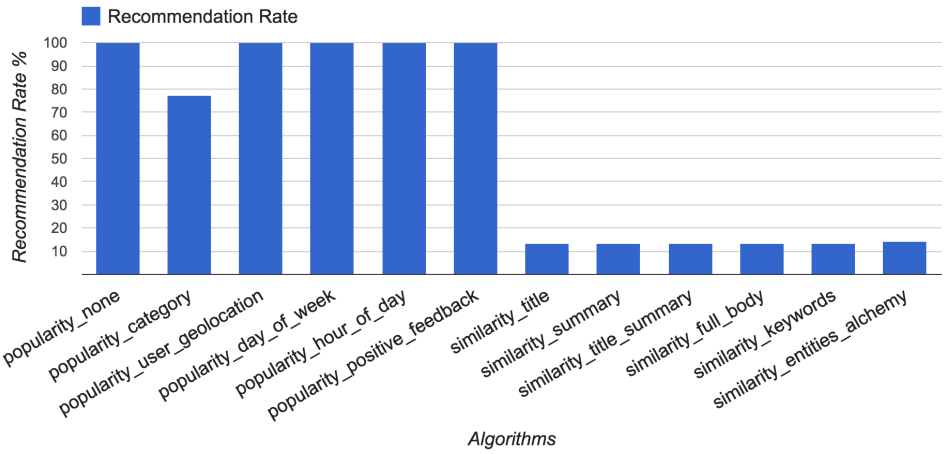

Fig. 2. The barchart gives an overview of the recommendation rate of the 12 algorithms employed. Recommendation rate is the percentage of times when an algorithm is capable of producing a recommendation.

7.3 Lessons Learned

8 Conclusions

References

1. ORP.

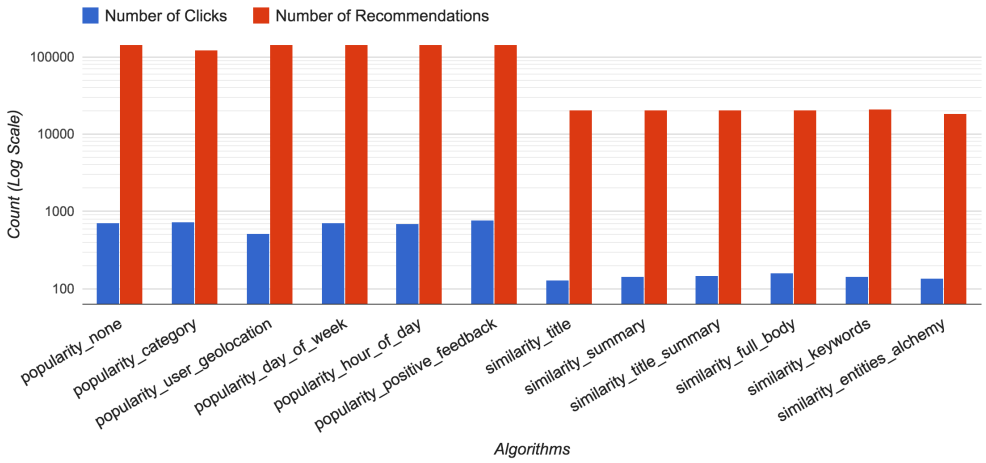

Fig. 3. A comparison between the number of recommendations an algorithms generated and the number of corresponding clicks it got.

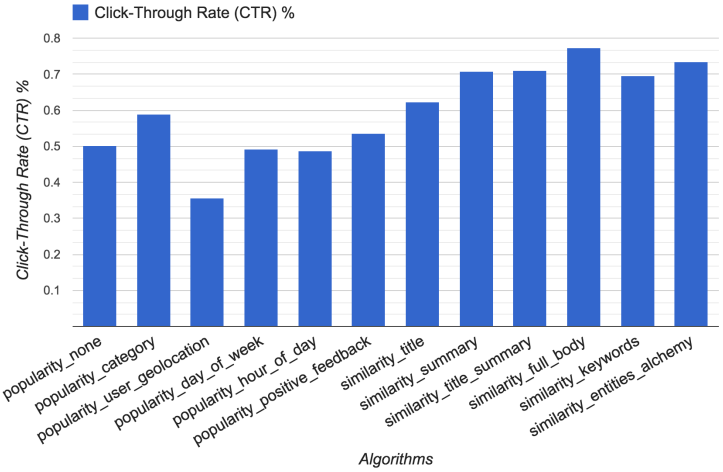

Fig. 4. An overview of the Click-Through Rate (CTR) performance of the algorithms. As we can see similarity algorithms perform better with a maximum CTR of 0.77% achieved by similarity-full-body.

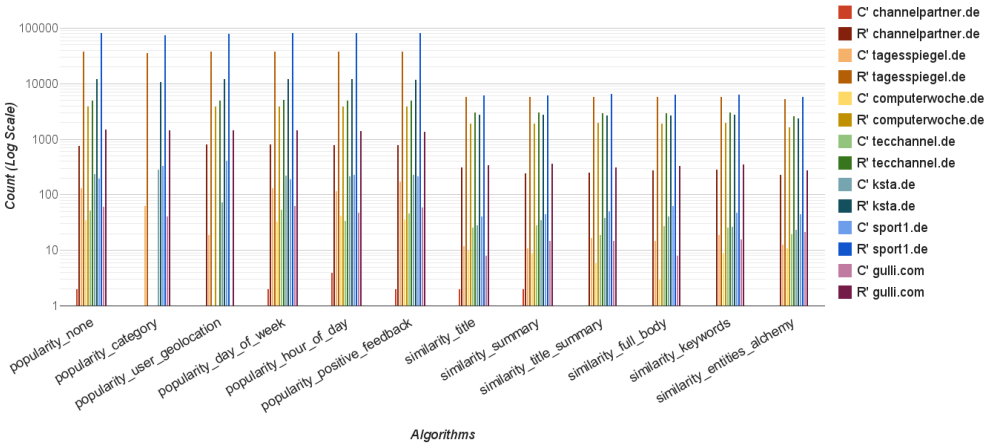

Fig. 5. Too much info on this graph should be substitured by the scatter plot posted on slack

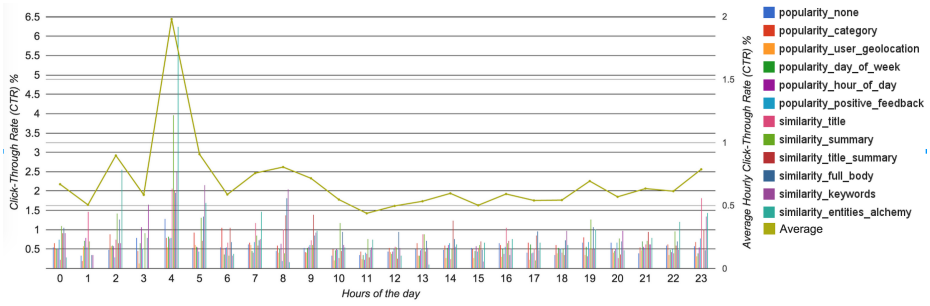

Fig. 6. We look at the Click-Through Rate (CTR) performance of algorithms, over the daily news cycle, and we see that at 4 o'clock people are much more responsive to similarity-based suggestions, meanig that their reading pattern at this hour is much more focused.

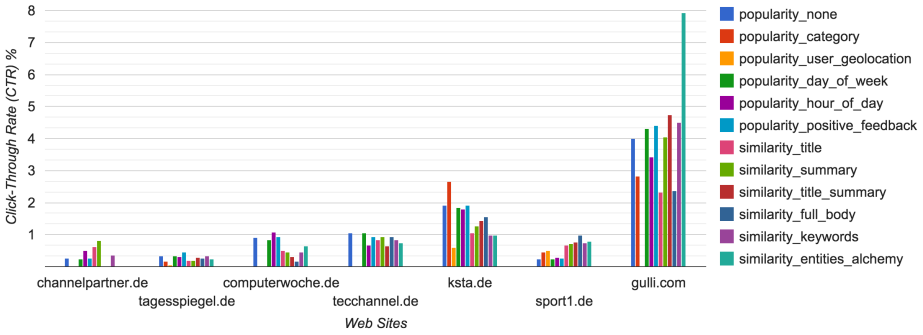

Fig. 7. Here we examine the algorithm's Click-Through Rate (CTR) by domains. There are two groups of two domains where ksta.de and gulli.com are in one, with an exceptionally good, relative to the rest, whereas channelpartner.de and tagesspiegel.de form the other group of poor performance.

2. G Adomavicius and A Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, May 2005.
3. D Agarwal, B C Chen, P Elango, and X Wang. Personalized click shaping through lagrangian duality for online recommendation. In *Proceedings of the 35th ...*, 2012.
4. M Capelle, F Frasincar, and M Moerland. Semantics-based news recommendation. In *Proceedings of the 2nd ...*, 2012.
5. A S Das, M Datar, A Garg, and S Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th ...*, 2007.
6. Blaž Fortuna, Carolina Fortuna, and Dunja Mladenić. Real-Time News Recommender System. *Machine Learning and Knowledge Discovery in Databases*, 6323(Chapter 38):583–586, 2010.
7. F Garcin, K Zhou, B Faltings, and V Schickel. Personalized news recommendation based on collaborative filtering. *... of the The 2012 IEEE/WIC/ ...*, 2012.
8. L Iaquinta, M de Gemmis, P Lops, G Semeraro, M Filannino, and P Molino. Introducing Serendipity in a Content-Based Recommender System. *Audio, Transactions of the IRE Professional Group on*, pages 168–173, September 2008.
9. I Ilievski and S Roy. Personalized news recommendation based on implicit feedback. In *Proceedings of the 2013 International News ...*, 2013.
10. B Kille, F Hopfgartner, T Brodt, and T Heintz. The plista dataset. In *Proceedings of the 2013 ...*, 2013.
11. L Li and T Li. News recommendation via hypergraph learning: encapsulation of user behavior and news content. In *Proceedings of the sixth ACM international conference ...*, 2013.
12. Yuanhua Lv, Taesup Moon, Pranam Kolari, Zhaohui Zheng, Xuanhui Wang, and Yi Chang. Learning to model relatedness for news recommendation. *the 20th international conference*, pages 57–66, March 2011.