Stephan Doerfel
Andreas Hotho
Robert Jäschke
Folke Mitzlaff
Juergen Mueller (Eds.)

20DC13
ECML PKDD Discovery Challenge

Recommending Given Names

International Workshop at
the European Conference on Machine Learning and
Principles and Practice of Knowledge Discovery in Databases
in Prague, Czech Republic, September 27th, 2013

Table of Contents

Preface

All over the world, future parents are facing the task of finding a suitable given name for their children. Their choice is usually influenced by a variety of factors, such as the social context, language, cultural background and especially personal taste. Although this task is omnipresent, little research has been conducted on the analysis and application of interrelations among given names from a data mining perspective.

Since 1999 the ECML PKDD embraces the tradition of organizing a Discovery Challenge, allowing researchers to develop and test algorithms for novel and real world datasets. The Discovery Challenge 2013[1] tackled the task of recommending given names in the context of the name search engine Nameling. It consisted of an *offline* and an *online* phase. In both phases, participants were asked to create a name recommendation algorithm that could provide suitable suggestions of given names to users of Nameling.

More than 40 participants/teams registered for the challenge, of which 17 handed in predictions of the offline challenge. After the end of the offline phase 6 teams submitted a paper. All papers have been peer reviewed and can be found in these proceedings. The different approaches to the challenge are presented at the ECML PKDD workshop on September 27th, 2013, in Prague, Czech Republic. The online challenge ran until the day before the workshop and four teams successfully participated with implementations meeting all required criteria. Details of the two challenge tasks, winners of both phases and an overview of the main findings are presented in the first paper of these proceedings.

The organizers would like to sincerely thank the challenge's sponsor Kasseler Sparkasse for donating the trophy money for the challenge's awards and the organizers of ECML PKDD 2013 for their support in the organization of the challenge and the workshop.

Kassel, October 2013
Stephan Doerfel, Andreas Hotho, Robert Jäschke,
Folke Mitzlaff, and Juergen Mueller

[1] `http://www.kde.cs.uni-kassel.de/ws/dc13/`

Summary of the 15th Discovery Challenge
Recommending Given Names

Folke Mitzlaff[1], Stephan Doerfel[1], Andreas Hotho[2,3], Robert Jäschke[3], and Juergen Mueller[1,3]

[1] University of Kassel, Knowledge Discovery and Data Engineering Group, Wilhelmshöher Allee 73, 34121 Kassel, Germany
{mitzlaff, doerfel, mueller}@cs.uni-kassel.de
[2] University of Würzburg, Data Mining and Information Retrieval Group, Am Hubland, 97074 Würzburg, Germany
hotho@informatik.uni-wuerzburg.de
[3] L3S Research Center, Appelstraße 4, 30167 Hannover, Germany
{hotho, juergen.mueller, jaeschke}@l3s.de

The 15th ECML PKDD Discovery Challenge centered around the recommendation of given names. Participants of the challenge implemented algorithms that were tested both offline – on data collected by the name search engine Nameling – and online within Nameling. Here, we describe both tasks in detail and discuss the publicly available datasets. We motivate and explain the chosen evaluation of the challenge, and we summarize the different approaches applied to the name recommendation tasks. Finally, we present the rankings and winners of the offline and the online phase.

1 Introduction

The choice of a given name is typically accompanied with an extensive search for the most suitable alternatives, at which many constraints apply. First of all, the social and cultural background determines, what a common name is and may additional imply certain habits, such as, e. g., the patronym. Additionally, most names bear a certain meaning or associations which, also depend on the cultural context. Whoever makes the decision is strongly influenced by personal taste and current trends within the social context. Either by preferring names which are currently popular, or by avoiding names which most likely will be common in the neighborhood.

Future parents are often aided by huge collections of given names which list several thousand names, ordered alphabetically or by popularity. To simplify and shorten this extensive approach, the name search engine Nameling (see Section 2) allows its users to query names and returns similar names. To determine similarity, Nameling utilizes Wikipedia's text corpus for interlinking names and the microblogging service Twitter for capturing current trends and popularity of given names. Nevertheless, the underlying rankings and thus the search results are statically bound to the underlying co-occurrence graph obtained from Wikipedia and thus not personalized. Since naming a child is a very personal

task, a name search engine can certainly profit from personalized name recommendation.

The task of 15th ECML PKDD Discovery Challenge was to create successful recommendation algorithms that would suggest suitable given names to future parents. The challenge relied on data gathered by Nameling and consisted of two phases, i. e., an *offline* and an *online* challenge. In both phases, participants were asked to provide a name recommendation algorithm to solve the task.

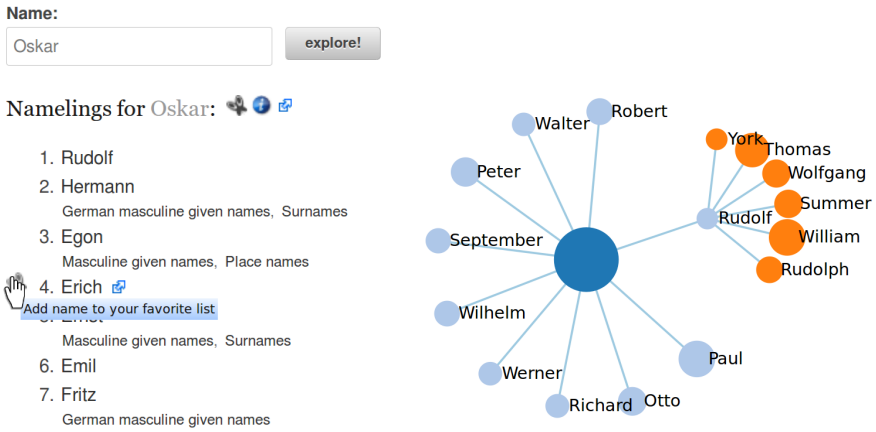
Task 1: The Offline Challenge. In the first phase, recommenders have been evaluated in an offline setting. To train an algorithm, the participants had been provided with a public training data set from Nameling's access logs, representing user search activities. Given a set of names for which a user had shown interest in, the recommender should provide suggestions for further names for that user. A second, private dataset from Nameling contained further search events from users of Nameling. To win the challenge, participants had to predict these search events. Details of the offline phase are discussed in Section 3 where we also summarize the different approaches to solve the challenge as well as the ranking of the participating teams and the challenge's winners.

Task 2: The Online Challenge. The online phase took place after Task 1 had been completed. The participants implemented a recommendation service that could be actively queried via HTTP and would provide names according to the participant's algorithm. These recommendation were shown to actual users of Nameling and their quality was measured by counting user's clicks on recommended names. We elaborate on the online challenge in Section 4.

2 The Name Search Engine Nameling

Nameling is designed as a search engine and recommendation system for given names. The basic principle is simple: The user enters a given name and gets a browsable list of relevant, related names, called "*namelings*". As an example, Figure 1a shows the namelings for the classical masculine German given name "Oskar". The list of namelings in this example ("Rudolf", "Hermann", "Egon", etc.) exclusively contains classical German masculine given names as well. Whenever an according article in Wikipedia exists, categories for the respective given name are displayed, as, e. g., "*Masculine given names*" and "*Place names*" for the given name "Egon". Via hyperlinks, the user can browse for namelings of each listed name or get a list of all names linked to a certain category in Wikipedia. Further background information for the query name is summarized in a corresponding details view, where, among others, popularity of the name in different language editions of Wikipedia as well as in Twitter is shown. As depicted in Figure 1b, the user may also explore the "neighborhood" of a given name, i. e., names which co-occur often with the query name.

From a user's perspective, Nameling is a tool for finding a suitable given name. Accordingly, names can easily be added to a personal list of favorite names. The list of favorite names is shown on every page in the Nameling and can be shared with a friend, for collaboratively finding a given name.



(a) Namelings (b) Co-occurring names

Fig. 1: A user query for the classical German given name "Oskar".

2.1 Computing Related Names

To generate the lists of related names, Nameling makes use of techniques that have become popular in the so called "Web 2.0". With the rise of the latter, various social applications for different domains – offering a huge source of information and giving insight into social interaction and personal attitudes – have emerged that make use of user generated data (e. g., user profiles and friendships in social networks or tagging data in bookmarking systems).

The basic idea behind Nameling was to discover relations among given names, based on such user generated data. In this section, we briefly summarize how data is collected and how relations among given names are established. Nameling is based on a comprehensive list of given names, which was initially manually collected, but then populated by user suggestions. Information about names and relations between them is gathered from three different data sources, as depicted in Figure 2:

Wikipedia: As basis for discovering relations among given names, co-occurrence graphs are generated for each language edition of Wikipedia separately. That is, for each language, a corresponding data set is downloaded from the Wikimedia Foundation[4]. Afterwards, for any pair of given names in our dataset, the number of sentences where they jointly occur is determined. Thus, an undirected graph is obtained for every language, where two names are adjacent if they occur together at least in one sentence within any of the articles and the edge's weight is given by the number of such sentences.

[4] "Database dump progress.", *Wikimedia*. 2012. `http://dumps.wikimedia.org/backup-index.html` (May 1, 2013)

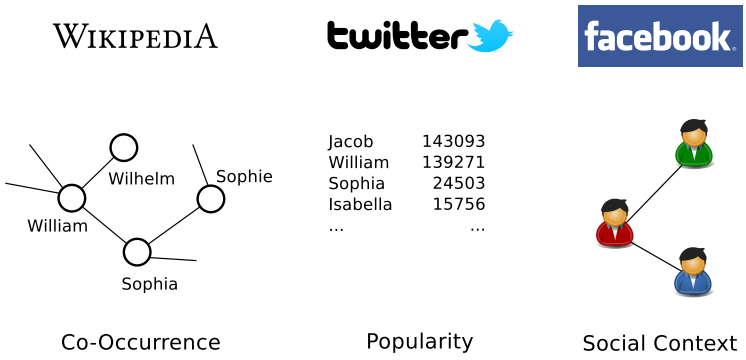


Fig. 2: Nameling determines similarities among given names based on co-occurrence networks from Wikipedia, popularity of given names via Twitter and social context of the querying user via Facebook.

Relations among given names are established by calculating a vertex similarity score between the corresponding nodes in the co-occurrence graph. Currently, namelings are calculated based on the cosine similarity (cf. [2]).

Twitter: For assessing up-to-date popularity of given names, a random sample of status messages in Twitter is constantly processed via the Twitter streaming API[5]. For each name, the number of tweets mentioning it is counted.

Facebook: Optionally, a user may connect Nameling with Facebook[6]. If the user allows Nameling to access his or her profile information, the given names of all contacts in Facebook are collected anonymously. Thus, a "social context" for the user's given name is recorded. Currently, the social context graph is too small for implementing features based on it, but it will be a valuable source for discovering and evaluating relations among given names.

2.2 Research around Nameling

Beside serving as a tool for parents-to-be, Nameling is a research platform too. The choice of a given name is influenced by many factors, ranging from cultural background and social environment to personal preference. Accordingly, the task of recommending given names is per se subject to interdisciplinary considerations.

Within Nameling, users are anonymously identified via a cookie that is, a small identification fragment which uniquely identifies a user's web browser. Although a single user might use several browsers or computers, Nameling uses the simple heuristic of treating cookies identification for users.

[5] Twitter Developers. `https://dev.twitter.com/docs/api/1/get/statuses/sample` (May 3, 2013)

[6] `https://facebook.com/`

The Nameling dataset arises from the requests that users make to Nameling. More than 60,000 users conducted more than 500,000 activities within the time range of consideration (March 6th, 2012 until February 12th, 2013). For every user, Nameling tracks the search history, favorite names and geographical location based on the user's IP address and the GeoIP[7] database. All these footprints together constitute a multi-mode network with multiple edge types. Analyzing this graph (or one of its projections) can reveal communities of users with similar search characteristics or cohesive groups of names, among others. In the context of the Discovery Challenge, the data of Nameling is used to train and to test given name recommendation algorithms.

3 Offline Challenge

The first task of the Discovery Challenge was to create an algorithm that produces given name recommendations – given a list of users with their previous history of name search requests to Nameling. The evaluation of these algorithms was conducted in a classical offline prediction scenario. A large dataset from Nameling was split into a public training dataset and a secret test dataset. In the following we describe details of the task and the dataset. In the second part of this section we summarize the participants' approaches and their results in the challenge.

3.1 Task

In the challenges, we deal with a standard binary item recommendation task. Users of the name search engine Nameling have expressed interest in certain names by searching for them or requesting their details. These interactions with the system are interpreted as (binary) positive feedback to these names, while there is no explicit negative feedback - only names towards which we do not know the user's attitude. A recommender algorithm must determine, which of these names will be of interest to the user.

Participants were given a public dataset to train their algorithms on. For the overall evaluation a second dataset containing only a list of users was given to them. The task in the offline challenge then was to produce for each user in that second dataset a list of 1,000 name recommendations, ordered by their relevance to the user at hand.

For the challenge no further restrictions were made regarding the choice of methodology or additional data. On the contrary, participants were encouraged to make use of any kind of data they might find suitable, e. g., family trees, location information, data from social networks, etc.

[7] "GeoIP databases and web services.", *MaxMind*. `http://www.maxmind.com/en/geolocation_landing` (May 3, 2013)

3.2 Challenge Data

For the challenge, we provided data from the name search engine Nameling, containing users together with their (partial) interaction history in Nameling. A user interaction hereby is always one of the following:

`ENTER_SEARCH` The user entered a name directly into Nameling's search field.
`LINK_SEARCH` The user followed a link on some result page (including pagination links in longer result lists).
`LINK_CATEGORY_SEARCH` Wherever available, names are categorized according to the corresponding Wikipedia articles. The users clicked on such a category link to obtain all accordingly categorized names.
`NAME_DETAILS` The user requested some detailed information for a name using a respective button.
`ADD_FAVORITE` The user added a name to his list of favorite names.

The full dataset contains interactions from Nameling's query logs, ranging from March 6th, 2012 to February 12th, 2013. It contains profile data for 60,922 users with 515,848 activities. This dataset was split into a *public training dataset* and a *secret test dataset*. For this purpose, a subset of users (in the following called test users) was selected for the evaluation. For each such test user, we withheld some of their most recent activities for testing according to the following rules:

- For each user, we selected the chronologically last two names for evaluation which had directly been entered into Nameling's search field (i.e., `ENTER_SEARCH` activity) and which are also contained in the list of known names. We thereby considered the respective time stamp of a name's first occurrence within the user's activities. We restricted the evaluation to `ENTER_SEARCH` activities, because all other user activities are biased towards the lists of names which were displayed by Nameling (see our corresponding analysis of the ranking performance in [2]).
- We considered only those names for evaluation which had not previously been added as a favorite name by the user.
- All remaining user activity after the (chronologically) first evaluation name has been discarded.
- We required at least three activities per user to remain in the data set.
- For previous publications, we already published part of Nameling's usage data. Only users not contained in this previously published data set, have been selected as test users.

With the above procedure we obtained two data sets[8]: The secret evaluation data set containing for each test user the two left out (i. e., `ENTER_SEARCH`) names and the public challenge data set containing the remaining user activities of the test users and the full lists of activities from all other users. The only

[8] Both datasets are available from the challenge's website: `http://www.kde.cs.uni-kassel.de/ws/dc13/downloads/`

applied preprocessing on our part was a conversion to lower case of all names. Additionally to the public training dataset, participants were provided with the list of all users in the test dataset to be used as input for their algorithms. Furthermore, we published a list of all names known to Nameling, which thus included all names occurring in the training or the test data (roughly 36,000 names).

3.3 Evaluation

Given the list of test users (see above), each participant produced a list of 1,000 recommended names[9] for each such user. These lists were then used to evaluate the quality of the algorithm by comparing for each test user the 1,000 names to the two left-out names from the secret test dataset. As usual, it is assumed that good algorithms will rank the left-out names high, since they represent the actual measurable interests of the user at hand.

The chosen assessment metric to compare the lists of recommendations is *mean average precision* (MAP@1000). MAP means to compute for each test user the precision at exactly the ranking positions of the two left-out names. These precision values are then first averaged per test user and finally in total to yield the score for the recommender at hand. While MAP usually can handle arbitrarily long lists of recommendations, for the challenge we restricted it to MAP@1000, meaning that only the first 1,000 positions of a list are considered. If one or both of the left out names were not among the top 1,000 name in the list, they were treated as if they were ranked at position 1,001 and 1,002 respectively. More formally, the score assigned to a participant's handed-in list is

$$\text{MAP@1000} := \frac{1}{|U|} \sum_{u=1}^{|U|} \left(\frac{1}{r_{1,u}} + \frac{2}{r_{2,u}} \right)$$

where U is the set of all test users, $r_{1,u}$ and $r_{2,u}$ are the ranks of two left-out names for user u from the secret evaluation dataset, and $r_{1,u} > r_{2,u}$.

The choice of the evaluation measure is crucial in the comparison of recommender algorithms. It is well-known that different measures often lead to different evaluation results and the choice of the metric must therefore be motivated by the use case at hand. In the case of Nameling, we had already seen that recommending given names is a difficult task [3]. For many users, many recommenders did not produce recommendation rankings with the test names among top positions. Thus, measures like precision@k – with k typically obtaining low values like 5 or 10 – make it hard to distinguish between results, especially for lower cut-off-thresholds k. MAP (Mean Average Precision) is a measure that is suitable for (arbitrarily long) ordered lists of recommendations. Like NDCG (Normalized Discounted Cumulative Gain) or AUC (Area Under the Curve) it evaluates the recommendations based on the positions of the left

[9] 1,000 name sound like a large number but given that parents currently read much longer and badly sorted name lists the number is reasonable (details below).

out items within the list of recommendations. It yields good scores when test items appear on the top positions of the recommendations and lower scores if they are ranked further below (unlike precision@k where lower ranked items are cut off and thus do not contribute to the score).

The main difference to AUC and NDCG is how the ranks of the left-out names are incorporated into the score. While AUC yields a linear combination of the two ranks, MAP takes a linear combination of the reciprocal ranks and NDCG a linear combinations of the (logarithmically) smoothed reciprocal ranks. Among these measures, MAP is the one that discriminates the strongest between higher or lower ranks and therefore was most suitable for the challenge.

Although we have just argued against cut-off-measures like precision@k it is reasonable to cut off lists at some point. In contrast to many other areas where recommendations are used (e.g., friend, movie, book, or website recommenders), in Nameling the time needed to evaluate a recommendation is very short: if you like a name, just click on it. Additionally, the cost in terms of money or time spent for following a recommendation that turns out bad, is very low. At the same time, finding the perfect name for a child is often a process of months rather than minutes (like for finding the next book to read or deciding which movie to watch) or seconds (deciding which tag to use or which website to visit on the net). Thus it is reasonable to assume that parents-to-be are willing to scroll through lists of names longer than the usual top ten – especially, considering that one of the traditional ways of searching for names is to go through first names dictionaries where names are listed unpersonalized, in alphabetical order. In such books usually there are a lot more than 1,000 names that have to be read and therefore it seems reasonable that readers of such books won't mind studying longer name lists on the web.

3.4 Summary of the Offline Challenge

Registered participants (or teams of participants) were given access to the public training dataset and the dataset containing the names of all test users. The offline challenge ran for about 17 weeks, beginning March 1st and ending July 1st, 2013. Every week, participants were invited to hand in lists of recommended names for the test users. These lists were evaluated (using the secret test dataset and the evaluation described above) to produce a weekly updated leaderboard. The leaderboard allowed the participants to check on the success of their methods and to compare themselves with the other participants. Since frequently updated results would constitute an opportunity for the participants to optimize their algorithms towards this feedback or even to reverse-engineer the correct test names, the leaderboard was not updated more often than once a week.

Participants and Winners More than 40 teams registered for the challenge of which 17 handed in lists of recommended names. Of these 17, six teams submitted a papers which are summarized below in Section 3.5. Table 1 shows the final scores of the 17 teams and reveals the winners of the offline phase:

1. place goes to team *uefs.br* for their approach using a features of names.
2. place is won by team *ibayer* using a syntactically enriched tensor factorization model.
3. place goes to team *all your base* for their algorithm exploiting a generalized form of association rules.

Table 1: The final results of the 17 teams in the offline challenge, showing all the participants' team names, together with the achieved MAP@1000 score.

Pos.	Team Name	MAP@1000
1	uefs.br	0,0491
2	ibayer	0,0472
3	all your base	0,0423
4	Labic	0,0379
5	cadejo	0,0367
6	disc	0,0340
7	Context	0,0321
8	TomFu	0,0309
9	Cibal	0,0262
10	thalesfc	0,0253
11	Prefix	0,0203
12	Gut_und_Guenstig	0,0169
13	TeamUFCG	0,0156
14	PwrInfZC	0,0130
15	persona-non-data	0,0043
16	erick_oliv	0,0021
17	Chanjo	0,0016

Figure 3 shows the scores of those six teams that handed in papers in time describing their approach. Only described approaches can be judge and presented at the workshop and therefore, all the other results are not considered in the remaining discussion. Additionally, two baselines (NameRank from [3] and the simple most-popular recommender) are presented in Figure 3. The latter simply suggests to any user those names that have been queried the most often in the past. It is thus unpersonalized and rather ad-hoc. NameRank is a variant of the popular personalized PageRank algorithm [1]. From the baseline results we can already tell that the recommendation problem is indeed hard, as the scores are rather low (between 0.025 and 0.030). On the other hand, we can observe that the simple most-popular is not that much worse than the much more sophisticated PageRank-like approach. The first approaches of almost all participants yielded scores lower or comparable to those of the baselines. However, over the course of the challenge the scores improved significantly and by the end of the challenge all teams had produced algorithms that outperformed both baselines.

To compare the recommenders' performances in greater detail, Figure 4 shows the cumulative distribution of the different ranking positions $(1, \ldots, 1000)$ for the

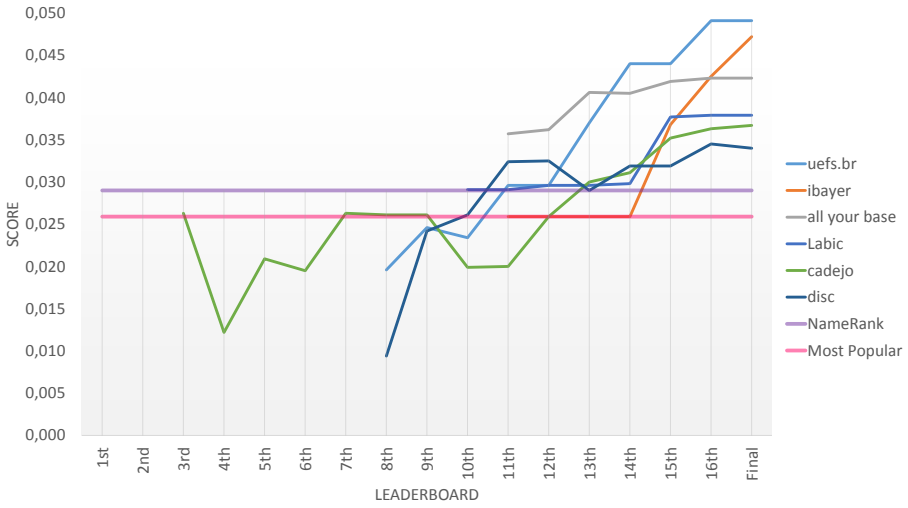


Fig. 3: The scores of the six teams that handed in papers plotted of the 17 weeks runtime of the offline challenge. For comparison, two baselines have been added (the two constant lines): NameRank and Most Popular.

top three algorithms. For each recommender system and every ranking position k, displayed is the number of hold-out names – out of the $8,280$ hold-out names from the $4,140$ test users with two hold-out names each – that had a rank smaller than or equal to k on the list of recommended names. We can observe, that the three curves are very close together, indicating that the distributions of ranks are similar. Comparing the results of the top placed two algorithms (team *uefs.br* and team *ibayer*), we see that the former has predicted more of the hold-out names in its top $1,000$ lists in total. However, the latter succeeded in placing more hold-out names among the top ranking positions ($k \leq 400$). The distribution of the third algorithm (team *all your base*) is almost identical with that of team *uefs.br* over the first 300 ranks, but then falls behind.

3.5 Approaches

In the offline phase of the challenge, six teams documented their approaches in the papers that are included in the challenges proceedings. In the following, the key idea of each approach is summarized. Using the respective team name of each paper's authors, their scores can be identified in Figure 3.

A mixed hybrid recommender system for given names
The paper by Rafael Glauber, Angelo Loula, and João B. Rocha-Junior (team *uefs.br*) presents a hybrid recommender which combines collaborative filtering, most popular, and content-based recommendations. In particular the latter contributes with two interesting approaches (Soundex and splitting

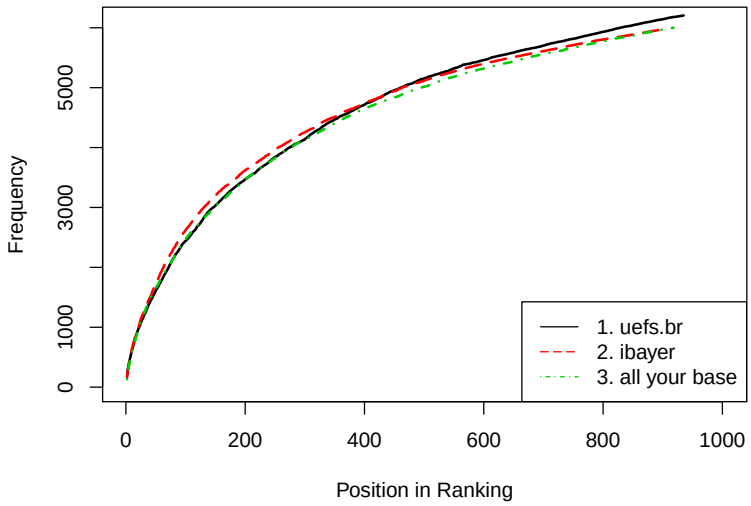


Fig. 4: Cumulative distribution of the ranking positions for the top three recommendation systems described in Section 3.5.

of invalid names) that are fitted to the problem at hand. The combination of the three approaches as a concatenation is likely the reason for the success in the challenge, yielding the highest score on the test dataset and thus winning the offline phase.

Collaborative Filtering Ensemble for Personalized Name Recommendation
Bernat Coma-Puig, Ernesto Diaz-Aviles, and Wolfgang Nejdl (team *cadejo*) present an algorithm based on the weighted rank ensemble of various collaborative filtering recommender systems. In particular, the classic item-to-item collaborative filtering is considered in different, specifically adopted variants (considering only `ENTER_SEARCH` activities vs. all activities, frequency biased name sampling vs. recency biased name sampling), item- and user-based CF as well as PageRank weights for filling up recommendation lists with less than 1,000 elements. The weights of the ensemble are determined in experiments and yield a recommender that outperforms each of its individual components.

Nameling Discovery Challenge - Collaborative Neighborhoods
The paper by Dirk Schäfer and Robin Senge (team *disc*) created an algorithm which combines user-based collaborative filtering with information about the geographical location of the users and their preference for male/female and long/short names. The paper further explores the use of two different similarity measures, namely Dunning and Jaccard, and find that the rather exotic one Dunning yields better recommendations than the Jaccard measure.

Improving the Recommendation of Given Names by Using Contextual Information

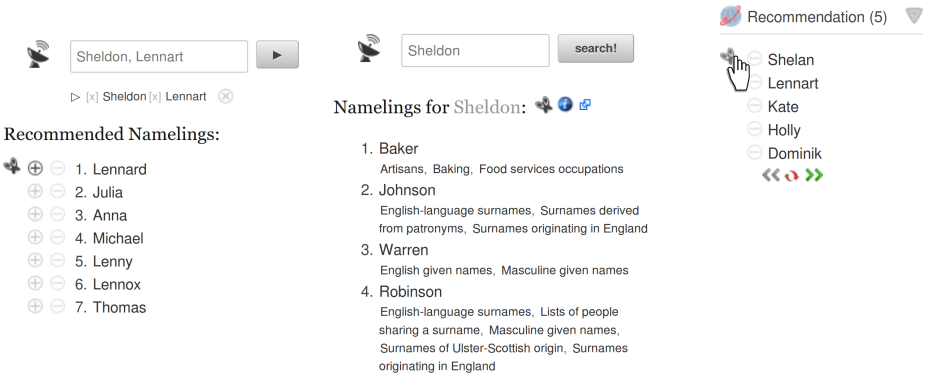
The paper by Marcos Aurélio Domingues, Ricardo Marcondes Marcacini, Solange Oliveira Rezende and Gustavo E. A. P. A. Batista (team *Labic*) presents two approaches to tackle the challenge of name recommendation: item-based collaborative filtering and association rules. In addition, the weight post filtering approach is leveraged to weight these two baseline recommenders by contextual information about time and location. Therefore, for each user-item pair the probability that the user accessed it at a certain context (i.e., time or location) is computed and used to weight the baseline results.

Similarity-weighted association rules for a name recommender system
The paper by Benjamin Letham (team *all your base*) considers association rules for recommendation. The key idea here is the introduction of an adjusted confidence value for association rules, capturing the idea of inducing a bias towards observations which stem from likeminded users (with respect to the querying user). This generalized definition of confidence is additionally combined with a previous approach of the author [4] which accounts for association rules with low support values, by adding in a Beta prior distribution. This recommender system achieved the third place in the challenge's offline task.

Factor Models for Recommending Given Names
The paper by Imannuel Bayer and Steffen Rendle (team *ibayer*) presents an approach using a sequential factor model that is enriched with syntactical name similarity – a prefix equality, called "*prefix smoothing*". The model is trained with a slight adoption of the standard Bayesian Personalized Ranking algorithm, while the final recommendation is obtained by averaging the rankings of different, independently trained models. This recommender system achieved the second place in the challenge's offline task.

4 Online Challenge

Conducting offline experiments is usually the first step to estimate the performance of different recommender systems. However, thus recommender systems are trained to predict those items that users found interesting without being assisted by a recommender system in the first place. In an online evaluation different recommenders are implemented into the running productive system. Their performance is compared in a test, where different users are assigned to different recommenders and the impact of the recommendations is estimated by analyzing the users responses to their recommendation. Like noted in [5] online experiments provide "the strongest evidence as to the true value of the system [...]", but have also an influence on the users as the recommendations are displayed before users decide where to navigate (click) next. In the challenge, the online phase gave the participants the opportunity to test the algorithms, they had created during the offline phase in Nameling. In the following we describe the setup that allowed the teams to integrate their algorithms with Nameling before we discuss this phase's results and winners.



(a) interactive recommendation

(b) sidebar recommendation

Fig. 5: Implemented recommendation use cases in Nameling: A user interactively queries for suitable name recommendations (a) or gets recommended names displayed in the sidebar of some regular Nameling page (b).

4.1 Recommendations in Nameling

The feature of recommendations in Nameling was introduced with the beginning of the challenge's online phase. To every page of the system was added a personalized list of recommended names. This list automatically adapts to the user's activities (e.g., the user's clicks or entering of favorite actions). Users may use this list to further search for names by clicking on one, add a name to his favorite names, or ban a name which they do not want to be recommended again. Additionally, users can visit an interactive recommendation site in Nameling, where they can enter names and will get personalized recommendations related to those names. The latter functionality is very similar to the usual results Nameling shows, the difference being that regular search results are non-personalized. Figure 5 shows how recommendations are displayed in Nameling's user interface.

To integrate their algorithms into Nameling, the participants had to implement a simple interface. The search engine itself provides a framework for the easy integration of recommender systems based on lightweight REST (HTTP + XML / JSON) interaction. Participants could choose to implement their recommender in Java or Python and to run their recommender in a web service on their own or to provide a JAR file to be deployed by the challenge organizers.

The recommender framework that integrates the different third-party recommender systems into Nameling is sketched in Figure 6. When a user of Nameling sends a request, a call for recommendations including the current user's ID is sent to each participating recommender. Recommendations are collected from each such recommender within a time frame of 500 ms, i. e., recommendations produced after that time are discarded. Using an equally distributed random

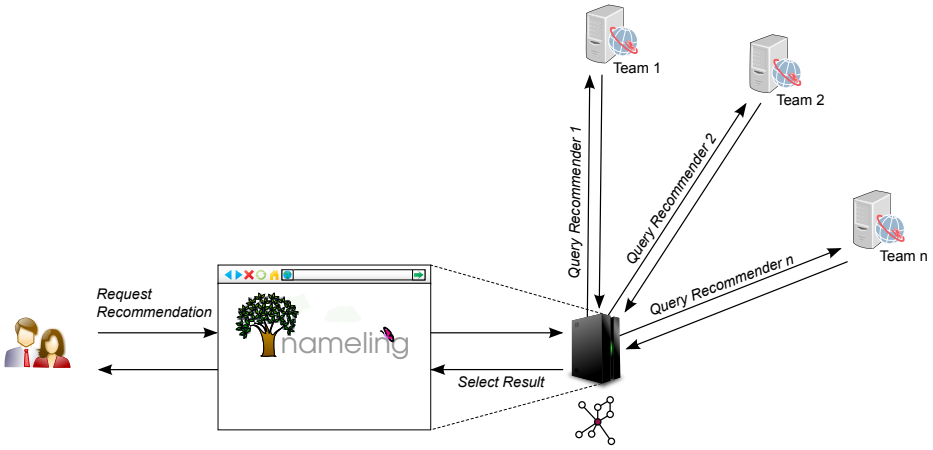


Fig. 6: Schematic representation of Nameling's online recommender framework.

function, one of the recommendation responses is selected and the recommended names are displayed to the current user in the response to their request. Once a recommender has been assigned to a user, this assignment is fixed for the duration of the current session unless the user specifically requests new recommendations.

4.2 Evaluation

Assessing the success of recommendations in a live system requires some quantity that can be measured and that represents the use of the recommendations to the user or to the system. Often used measures include a rise in revenue (e. g., for product recommendations), click counts on the recommended items, or comparisons to ratings that users assigned to recommended items. In the case of name recommendations, no particular revenue is created for the system as there are no products to be sold. Thus to evaluate different recommenders we focused on the interest that users showed in the recommended names. For the challenge we estimated this interest by the combined number of requests users made responding to the recommendations. More precisely, we counted all interactions that could be made on the recommender user interface (i. e., `LINK_SEARCH`, `LINK_CATEGORY_SEARCH`, and `ADD_FAVORITE` events). Here, we excluded the previously mentioned option to ban names as their interpretations is unclear. On the one hand, banning a name is certainly a negative response to that particular recommendation. On the other hand, since users are not bound to react to the recommendations at all, it is a clear sign of interest in the recommendations and could well be interpreted as a deselection of one uninteresting name among from a set of otherwise interesting names. Since the recommenders were assigned to different users using an equally distributed random function, the final measure

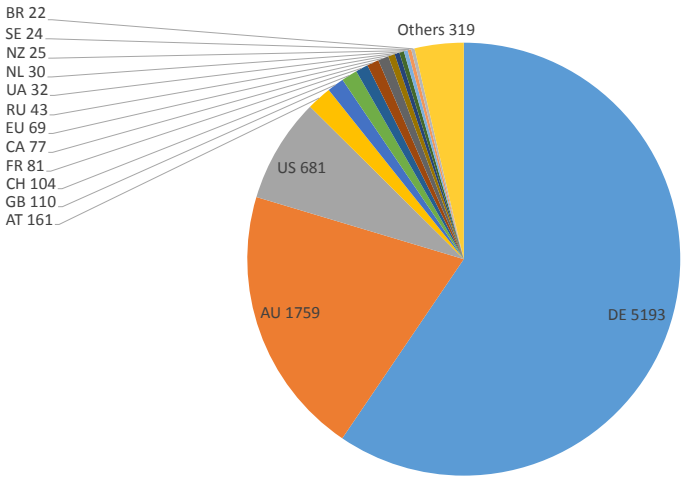


Fig. 7: The countries (according to IP address) of Nameling's visitors during the online phase of the challenge.

was simply the sum of all considered requests in one of the three mentioned categories.

4.3 Summary of the Online Challenge

The online phase ran from August, 1st to September 24th, 2013. During the time of the online phase, more than 8,000 users visited Nameling engaging in more than 200,000 activities there. Figure 7 shows the distributions of the users over their countries (it is to be expected, the home country is an important influence on the choice of names). While most of the requests came from Germany, followed by Austria, the largest number of visitors from an English speaking country came from the US.

Participants and Winners Of the teams that contributed to the challenge proceedings, five entered their algorithms in the online challenge. Of those five teams, four managed to produce recommendations within the time-window of 500 ms: "all your base", "Contest", "ibayer", and "uefs.br". Figure 8 shows for each of these four teams the number of responses – in terms of clicks to one of three categories of links related to the recommended names (see Section 4.2 – to their recommendations. The clear winner of the online phase is team "ibayer": Immanuel Bayer and Steffen Rendle (Paper: Factor Models for Recommending Given Names). Ranks two and three go to teams "all your base" and "uefs.br" respectively. Compared to the offline challenge, we find, the three top teams of the offline phase were the same that constituted the top three of the online

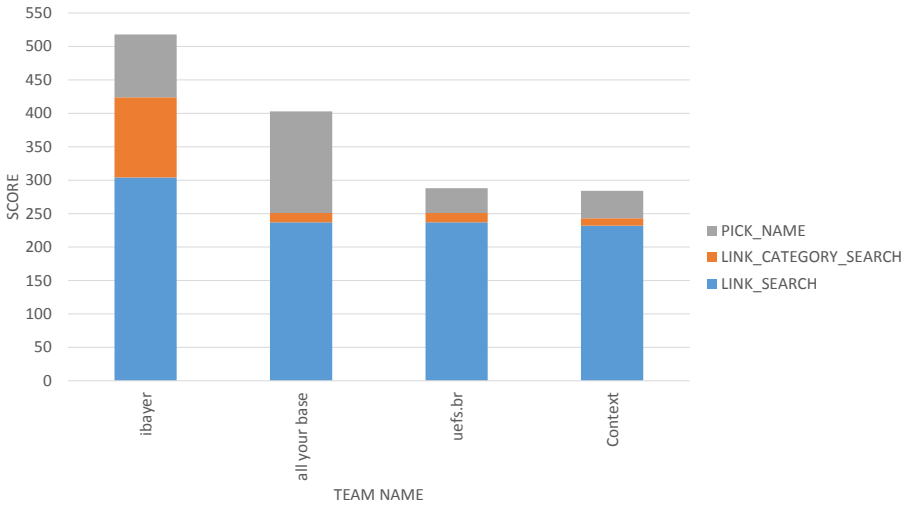


Fig. 8: Schematic representation of Nameling's online recommender framework.

phase. It is however interesting to note that the order of the teams changed – team "uefs.br" fell from rank one to rank three. It is also worth noting that although team "Context" yielded a MAP@1000 score of only 0,0321 in the online challenge, compared to team "uefs.br" with 0,0491, both teams were almost equally successful in the online phase. It thus seems that the offline testing has indeed been a reasonable precursor, yet also that the offline scenario does not fully capture the actual use case.

5 Conclusion

The 15th Discovery Challenge of the ECML PKDD posed the task of recommending given names to users of a name search engine. In the two parts of the challenge, the offline and the online phase several teams of scientists implemented and augmented recommendation algorithms to tackle that problem. In their approaches, participants mainly chose to use well-established techniques like collaborative filtering, tensor factorization, popularity measures, or association rules and hybridization thereof. Participants adapted such algorithms to the particular domain of given names exploiting name feature like gender, a name's prefix, a name's string length, or phonetic similarity. In the offline challenge, six teams entered their approaches and by the end of the phase, each team had produced a new algorithm outperforming the previously most successful recommender NameRank. The achieved scores of the individual recommenders were yet rather low (compared to other domains were recommenders are applied). This shows that there is yet much to be explored to better understand and predict the attitude of users towards different names. Through the challenge, a multitude of

ideas and approaches has been proposed and a straight forward next step will be to explore their value in hybrid recommender algorithms. Hybridization has been used already by several participants with great success.

The online challenge opened the productively running name search engine Nameling to the scientific community, offering the possibility to implement and test name recommendation algorithms in a live system. Results showed that the actual performance varied from that measured in the offline challenge. However, it could also be observed that despite the low scores in the offline phase, the recommendations were perceived by users and were able to attract their attention.

As organizers, we would like to thank all participants for their valuable contributions and ideas.

References

1. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
2. F. Mitzlaff and G. Stumme. Relatedness of given names. *Human Journal*, 1(4):205–217, 2012.
3. F. Mitzlaff and G. Stumme. Recommending given names, 2013. cite arxiv:1302.4412Comment: Baseline results for the ECML PKDD Discovery Challenge 2013.
4. C. Rudin, B. Letham, A. Salleb-Aouissi, E. Kogan, and D. Madigan. Sequential event prediction with association rules. *COLT 2011 - 24th Annual Conference on Learning Theory*, 2011.
5. G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011.

A mixed hybrid recommender system for given names

Rafael Glauber[1], Angelo Loula[1], and João B. Rocha-Junior[2]

[1] Intelligent and Cognitive Systems Lab (LASIC)
[2] Advanced Data Management Research Group (ADaM)
State University of Feira de Santana (UEFS)
Feira de Santana, Bahia, Brazil
{rafaelglauber, angelocl, joao}@ecomp.uefs.br

Abstract. Recommender systems are data filtering systems that suggest data items of interest by predicting user preferences. In this paper, we describe the recommender system developed by the team named *uefs.br* for the offline competition of the *15th ECML PKDD Discovery Challenge 2013* on building a recommendation system for given names. The proposed system is a hybrid recommender system that applied a content-based approach, a collaborative filtering approach and a popularity approach. The final recommendation is composed by the results of these three different approaches, in which parameters where optimized according to experiments conducted in datasets built from train data.

1 Introduction

Choosing a given name can be a hard task, considering the large amount of available names and the diverse personal preferences and cultural contexts [4]. The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD 2013, organized a Discovery Challenge on building a recommendation system for given names. The challenge has two phases, an offline competition, for predicting search activity for users based on data from the nameling.net website, and an online competition where teams would integrate their recommender system into the nameling.net website.

Here we describe the recommender system developed for the offline competition by the team named *uefs.br* (initially called *sertão*). The uefs.br team was ranked first in the last public leaderboard of the offline competition, with a score of 0,0491. The proposed system was a hybrid recommender system that applied a content-based approach, a collaborative filtering approach and a popularity approach.

In summary, the main contributions of this paper concern proposing a popularity-based, content-based and collaborative-based recommender system; proposing a mixed hybrid recommender system based on recommendations of the previous systems; and evaluating the performance of the hybrid recommender system using different settings.

The rest of this paper is organized as follows. In the next section, we present an overview on recommender systems. Section 3 states the proposed offline challenge task and how recommendations where evaluated. Then, we describe our proposed hybrid recommender system in Section 4. In Section 5, experimental results are presented along with parameters settings and the datasets used, and the last section gives final remarks on the proposed solution.

2 Recommender Systems

A Recommender System (RS) is a filtering system that suggests items of possible interest for a given user [6]. The problem of recommendation can be seen as the problem of estimating the user rating for items not yet rated by the user. Items predicted with high rating for a given user can be offered by the system as a recommendation.

There are two main approaches for recommendation, Content-based and Collaborative Filtering. Different recommendation approaches can be combined in a third approach, a Hybrid [1] one.

Content-based Approach. The Content-based (CB) approach [5] recommends new items according to their content similarity with items of previous user interest. It has its basis in Information Retrieval (IR) and Machine Learning (ML) [6, 3, 8]. This approach employs, for example, content representation and comparison techniques from IR besides classification algorithms from ML to represent items previously assessed by the user and compare with other items, in order to recommend similar items.

Collaborative Filtering Approach. Instead of comparing items, the Collaborative Filtering (CF) approach [7] compares users based on their item interests, and recommends new items that were of interest to such similar users. This technique, called the "automation of word-of-mouth" by Shardanand and Maes [9], has a filtering principle in which the workload of identifying relevant content (or quality from the perspective of an individual user) is partitioned among system users, who record their personal rating for retrieved items.

Hybrid Approach. The previous approaches, and any other approach, can be combined in order to employ complementary aspects of each other [1, 2]. Recommender systems that employ more than one approach are named a hybrid recommender systems. The main challenge of this approach is how to combine approaches, for example, by combining items suggested by a collaborative filtering approach and a content-based approach, in order to suggest items similar to those previously "recommended" by the active users and items of interest that are "recommended" by users with similar taste.

3 Problem Statement

The offline competition phase from *15th ECML PKDD Discovery Challenge 2013* defined the task of predicting future searches for users of the nameling.net

website. The nameling.net website is a search engine and a recommendation system for given names, based on data observations from the social web [4]. As the user enters the website, he enters a given name and gets a browsable list of "relevant" names, called "namelings". Each name listed, including the searched one, has details associated with it, including Wikipedia articles (categories), popularity in Twitter and Wikipedia and names commonly co-occuring.

Data captured from the nameling.net website was provided for offline competition, regarding logged interactions from the website users, called activities. Every user activity is supplied with the (obfuscated) user ID, the type of activity, the associated name and a time stamp. The types of activities could be ENTER_SEARCH, when the user entered a name into the search field, LINK_SEARCH, when the user clicked a link on a result page, LINK_CATEGORY_SEARCH, when the user clicked on a wikipedia category related with a name, resulting in a list of all name in the same category, NAME_DETAILS, when the user gets details associated with a name, or ADD_FAVORITE, when the user adds a name to his list of favorite names.

A subset of users is selected as test users, and for each user in this subset, the last two ENTER_SEARCH activities were removed from his activity history. Each competition team is challenged to build a recommender system that provides an ordered list of names (up to 1000 names) for each test user that should include the names related to the two omitted activities. These two names are present in a provided list of nameling.net known names, but they may not occur as ENTER_SEARCH or ADD_FAVORITE in the given user provided history. This procedure provides two datasets, a secret evaluation dataset, with withhold names for test users, and a public challenge dataset, with all remaining user activities.

To evaluate the recommended list of names for each test user, the Mean Average Precision at position 1000 (MAP@1000) was used as the challenge score. Given a user i and his first omitted name found at position n_1 and the second one at position n_2, the Average Precision $AveP@1000_i$ and $MAP@1000$ for N users were given by:

$$AveP@1000_i = \frac{1}{2}.\left(\frac{1}{n_1} + \frac{2}{n_2}\right)$$

$$MAP@1000 = \frac{1}{N}.\sum_{i=1}^{N} AveP@1000_i$$

If one of the omitted names was not in the ordered list, n_2 was admitted to be at position 1001, and if both were not in the list, n_1 was set to 1001 and n_2, 1002.

4 Proposed Solution

To address the offline competition task, we built a hybrid recommender system that combines recommendations coming from different recommender systems, a

content-based one, a collaborative filtering one and a popularity based one. In Section 4.1, we present the popularity-based approach that recommends names based on the popularity of the names in the collection. In Section 4.2, we describe our collaborative filtering approach that retrieves given names from users with similar name interests. In Section 4.3, we describe our content-based approach that recommends new names based on phonetic string matching with names in the last activities of the given user. Finally, in Section 4.4, we present our hybrid approach that combines the previous techniques for recommending names.

4.1 Popularity-based approach

The popularity-based approach recommends popular (frequent) names. The idea is ranking names based on frequency in the collection and suggesting the same result ordered list to every test user, but removing from the recommendation list names that have previously being associated with an ENTER_SEARCH or ADD_FAVORITE activity of the given user.

In order to determine name popularity, we parse the public challenge dataset and, for each name we count the number of users that have such a name in their history of activities. If a name appears more than once in the activities of a given user, only one occurrence of the name is counted.

The big advantage of this approach is its simplicity and capacity for filling the list of one thousand names. The big disadvantage is that it is not customized for each user.

4.2 Collaborative Filtering approach

The collaborative filtering system produces a customized list of recommended names for each test user coming from names in the list of activity of similar users (neighbors). To determine neighbors for a given test user, a similarity measure is computed between this test user and all other users. Those users with the similarity measure above a given threshold are considered neighbors of the test user. The list of names recommended to the given test user is composed by the names appearing in the neighbors' list of activities and that are not in the given test user list as an ENTER_SEARCH and ADD_FAVORITE activity.

Computing the similarity between two users (neighborhood similarity). The neighborhood similarity measure between two users is computed taking into account only the valid (known) names associated with ENTER_SEARCH and LINK_SEARCH activities for both users, defining a user profile. Profiles are represented as a n-dimensional binary vector with each dimension representing a name among the n valid names. If a name occurs in a profile, the profile vector has value 1 in that dimension, otherwise, it has value 0. As neighborhood similarity measure, we use the well-known *cosine* similarity [3] between profile vectors of a test user T and another user U:

$$similarity(T, U) = cos(\theta) = \frac{\sum_{i=1}^{n} T_i \cdot U_i}{\sqrt{\sum_{i=1}^{n} T_i^2} \cdot \sqrt{\sum_{i=1}^{n} U_i^2}} \quad (1)$$

where n is number of valid names (vector size), T_i and U_i are the i-th dimension value in vector T and U, respectively.

Selecting the neighbors. The neighbors of a test user T (test user) are those users U whose cosine similarity is greater or equal a given *Similarity Threshold*.

Selecting the names. After selecting neighbors, all names present in these neighbors profiles compose a list of candidate names for recommendation. For each name, the score is the sum of the cosine similarity of the neighbors that contain the name. This score is used to rank the candidate names and order the list. The final list of names recommended by the Collaborative Filtering Approach is limited in the hybrid system (*Collaborative Filtering Limit*).

The advantage of this approach is that it suggests names from users of similar taste, producing a customized list of names for each user. The limitation is that it may not find enough neighbors (or no neighbors at all) and it may not be able to provide a list of one thousand names.

4.3 Content-based approach

Our content-based approach checks the names in the last activities of the test user in order to recommend similar names. The recommended names are selected from a list of candidate names, which are determined comparing the names in the last activities with the list of valid names using an algorithm for *phonetic string matching* [10]. This algorithm can be used to suggest spelling variations of a given name. For example, given the name "johane" and a list of valid names, this procedure can suggest "johan", "johanie", and "johanne". In our proposed system, we have employed the Soundex algorithm [10], but any other phonetic string matching algorithm could have been employed.

Soundex employs a parameter to define the minimum string similarity between two names and only names above this string similarity are considered. Therefore, if we increase the parameter, less names are returned by content-based recommendation, once only strictly more similar names are recommended. The number of names provided by this approach depends not only on the string similarity parameter but also on the specific name under consideration. Since names are compared to the list of valid names, some names may have no similar names (empty set), while others have more than 10 similar ones. Besides, we also need to order the list of similar names and, to that end, we rank the suggested names by popularity (Section 4.1), hence we can select the most popular names suggested by the content-based approach.

The content-based approach is divided into two phases. In the first phase, we find the candidate names and in the second phase, we select the best names for recommendation.

Finding candidate names (first phase). In order to find the candidate names, we employ three techniques.

- **First.** We check if the user's last activity is an ENTER_SEARCH activity associated with a valid name (a name in known names list), and insert all similar names (using Soundex) in the candidates names list.

- **Second.** We check if the user's last activity is a LINK_SEARCH, inserting it as candidate. This name, when it appears, is always the first candidate.
- **Third.** We check the last four activities of a test user, looking for invalid names (names not in known names list) associated with ENTER_SEARCH activities. The invalid names are then split using the following delimiters: " -#+,". The aim is to transform invalid names such as "stine#" in valid names such as "stine" or invalid names such as "Christina Alexandra" in two valid names "Christina" and "Alexandra". The valid names are added to the candidate names list. If, after name split, the result names are still invalid, we search for similar valid names using Soundex, which are also added as candidate names.

Selecting the best candidate names (second phase). The list of candidate names produced in the first phase can contain many names. Therefore, in order to select the best names, we rank the list of candidates by popularity (Section 4.1). In a pure content-based approach, we can select up to one thousand names; while in our hybrid approach, we limited the number of names according to a parameter *Content-based Limit*.

The main limitation of this approach is that, in most cases, it does not fill the list of one thousand names. Besides, it does not recommend names to users with last activities associated with names with great phonetic difference to all other known names. The big advantage is exploiting the names in the user's last activities for recommendation, once the challenge task is to predict two subsequent names in the user profile, which are highly temporally dependent on the last activities.

4.4 Hybrid approach

All the approaches proposed above have strong and weak points and, therefore, they may not present the best results when considered individually. However, if the list of recommended names is built by combining names obtained from all the previous approaches, we can have a hybrid recommender system with better results. In our proposal, the final list of recommended names is composed by the resulting list of recommended names from each of the three different approaches, characterized as a mixed hybrid system [1].

In order to find the best way to combine the results from the different approaches, we have tried different combinations of the previous approaches. Figure 1 presents the best combination found, where the two first items are obtained using the content-based approach, the next names (up to three hundred) are obtained using the collaborative filtering approach, and the remaining names are obtained using the popularity-based approach.

The rationale behind the hybrid approach is putting first the names related to the last few activities of the user (content-based approach), which is a short list that exploits high scoring positions in the MAP score, but leaves a lot of subsequent score positions for the following approach. Then, collaborative filtering fills the 300 following names in the recommendation list, which is still a

Fig. 1. Hybrid approach and its final combination.

user customized recommendation. Finally, to complete the list of 1000 names, since the previous two may not do so, we add popular names with overall high probability of being of interest.

5 Experimental Evaluation

An experimental evaluation was conducted to compare the different approaches proposed and the impact of parameters in the hybrid recommender system. In each experiment, we vary one parameter, while the others are fixed, therefore, showing the impact of one parameter in system results.

5.1 Settings

Our hybrid recommender system combines different approaches for building a recommendation list. Table 1 presents the parameters employed in the experimental evaluation, the default values are presented in bold.

Parameters	**Values**
Content-based limit	1, **2**, 3, 4, 5
Collaborative filtering limit	100, 200, **300**, 400, 500
Soundex parameter for valid names	0.93, 0.94, 0.95, **0.96**, 0.97
Soundex parameter for invalid names	0.89, 0.90, **0.91**, 0.92, 0.93
Similarity threshold	0.09, 0.10, **0.11**, 0.12, 0.13
Dataset	1, 2, 3

Table 1. Parameter settings used in the experiments.

Content-based Limit (ContentLimit) is the parameter that controls the maximum number of names recommended by the content-based approach.

Collaborative Filtering Limit (CollaborativeLimit) is the parameter that controls the maximum number of names recommended by the collaborative filtering approach.

Soundex Parameter for Valid Names (SoundexValid) and Soundex Parameter for Invalid Names (SoundexInvalid) defines minimum string similarity required for phonetically similarity between two names.

Similarity Threshold (SimThreshold) defines minimum cosine similarity required for a user to be considered neighbor of the test user (active user).

5.2 Datasets

In order to evaluate the impact of each parameter in our recommender system, we have created three evaluation datasets from the public challenge dataset. The datasets were created using the provided script to generate test users, then splitting the result test users dataset into three smaller datasets and combined with users from the public challenge dataset, keeping similar characteristics to the original dataset . In each new dataset, only test users have their profile changed (the last two ENTER_SEARCH activities was removed), while the profile of the other users are kept the same as in the public challenge dataset.

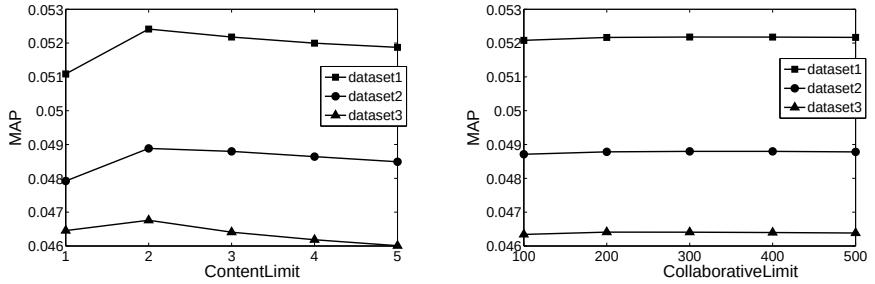
Properties	**Dataset 0**	**Dataset 1**	**Dataset 2**	**Dataset 3**
#valid names	17457	17326	17309	17291
#invalid names	14638	14232	14277	14175
#test users	4140	4139	4141	4728
avg #valid names per user	4.2643	4.1013	4.1017	4.0765
avg #invalid names per user	0.3087	0.2985	0.3000	0.2976
avg #users per valid name	14.8046	14.3463	14.3619	14.2884
avg #users per invalid name	1.2781	1.2714	1.2734	1.2725
max #valid names per user	1476	1476	1476	1476
max #invalid names per user	61	60	61	61
max #users per valid name	2263	2183	2189	2181
max #users per invalid name	57	54	55	56

Table 2. Datasets characteristics.

Table 2 shows the characteristics of the public challenge dataset (Dataset 0) and the three new datasets created (Datasets 1, 2, and 3). These characteristics were extracted from all activities, except LINK_CATEGORY_SEARCH. Each new dataset has a distinct set of test users. The valid names are those that are present in the provided list of known names, while the invalid names are ENTER_SEARCH activities whose names are unknown names.

5.3 Defining the number of items for each approach

As an initial step, we study the maximum number of names taken from each approach in our hybrid recommender system (Figure 2). The list of names recommended by the hybrid recommender system is composed by items obtained using the content-based approach, followed by items obtained using the collaborative filtering approach and, filling the rest of the list, with items obtained using the popularity-based approach. First, we study the limits of the content-based approach, then we study the limits of the collaborative-based approach.



(a) Content-based limit (b) Collaborative filtering limit

Fig. 2. Number of items from the (a) content-based and (b) collaborative filtering approaches used in the hybrid recommender system.

Content-based limit (ContentLimit). Figure 2(a) presents the MAP obtained with our hybrid recommender system, while the *ContentLimit* varies[3]. The best result was obtained using only two items from the content-based approach. Besides the small number of items, the impact of the content-based approach in our recommender system is significant, since the items recommended are the two first items in the list (Section 3).

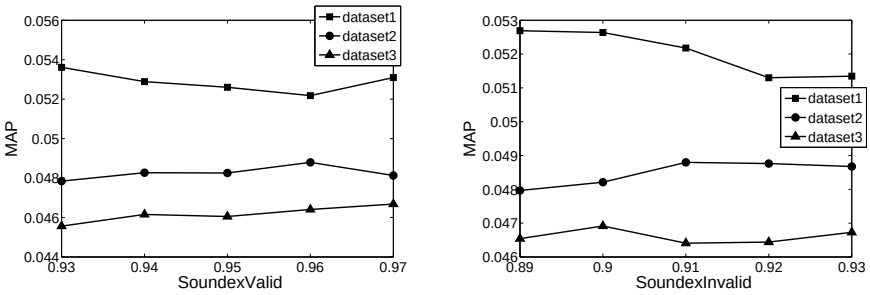
Collaborative filtering limit (CollaborativeLimit). Figure 2(b) presents the MAP obtained with our hybrid recommender system, while the *CollaborativeLimit* varies. The *CollaborativeLimit* includes the items obtained using the content-based approach. Therefore, when $CollaborativeLimit = 300$, it means that the two first items are obtained using the content-based approach, while the other 298 items are obtained using the collaborative-based approach. Results show a really small difference in MAP for variations of *CollaborativeLimit*, but a slightly better MAP for the value 300, which is the reason for using this value. This small difference also evidences that the collaborative filtering and the popularity approach almost compensates each other in MAP score, when more names from one approach and less from the other are included.

5.4 Calibrating the phonetic string matching algorithm

The impact of the phonetic string matching algorithm (Soundex) in our hybrid recommender system was also evaluated. The Soundex algorithm compares phonetic similar names and allows to locate known names similar to valid and invalid names (Section 4.3). Therefore, we have configured Soundex differently, depending whether the name is valid or invalid.

Figure 3(a) presents the MAP while varying the phonetic string similarity threshold for valid names, while Figure 3(b) shows the MAP for invalid names. Although there is no clear convergence of this parameters in all three datasets, we have chosen Dataset 2 as our benchmark as it approaches better results obtained (weekly) from the secret evaluation dataset and it also represents a

[3] In all figures, we employ the default values presented in Table 1, while varying one single parameter.



(a) Similarity for valid name (b) Similarity for invalid name

Fig. 3. String similarity threshold parameter in Soundex algorithm.

balance between the higher results obtained for Dataset 1 and lower results for Dataset 3. Therefore, we employed the value 0.96 for valid names and 0.91, which maximizes the MAP for Dataset 2.

5.5 Neighborhood selection

Another evaluation concerned the minimum similarity required for considering a user to be a neighbor of a test user in the collaborative filtering approach (Section 4.2). Only users whose similarity to a given test user is above this similarity threshold are considered neighbors, who can recommend names to the test user. Figure 4 presents the impact on the MAP for the similarity threshold varying from 0.09 to 0.13. The best results are obtained when this value is set to 0.11.

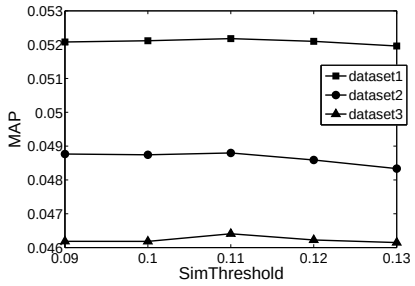


Fig. 4. The lowest degree of similarity required to be neighbor.

5.6 Comparing the different approaches individually

To evaluate the potential of each recommendation approach, we studied each one as an isolated recommender system. Figure 5 presents the MAP score for each of the three approaches, when they are considered individually. In these experiments, we fill the list with items recommended by a single approach, without limiting the number of items suggested. For example, for the content-based approach, we list all names that can be recommended by this approach without limiting to only two names.

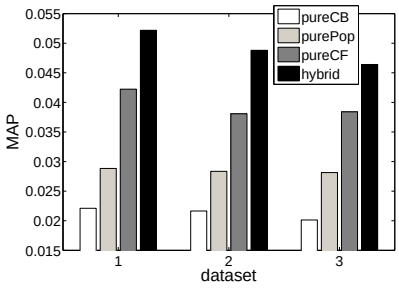


Fig. 5. Comparing approaches as isolated recommenders.

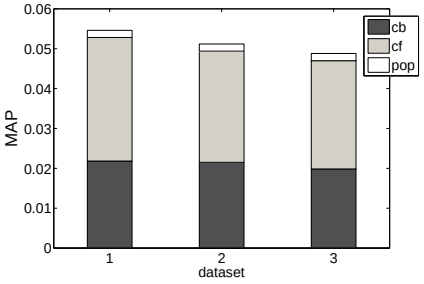


Fig. 6. Impact of each approach in the hybrid recommendation.

When considered individually as a pure recommender system, the collaborative filtering approach (pureCF) presents the best results, while the content-based approach (pureCB) is the one that presents the worst results, with popularity-based (purePop) having intermediate results. The content-based approach does not recommend many names, therefore, most lists of recommended names using this approach have much less than one thousand items (or even no recommendation at all). But pureCB has a considerable MAP score for such a limited list of names. Besides the good results obtained with the pure collaborative-based approach, the hybrid is able to combine characteristics from each approach producing the overall best results. These experiments shows the efficacy of our hybrid recommender system in aggregating recommendations from the different approaches.

5.7 Studying the impact of each approach to the hybrid system

In order to assess the individual contribution from each recommendation approach to the hybrid approach, we evaluate the MAP score composition from each one in the final recommendation list. Figure 6 presents the results obtained using the hybrid approach, showing the contribution of each approach in the result (cf:collaborative filtering, cb:content-based and pop:popularity-based). In this experiment, we employ the default values presented in Table 1.

The first thing to notice is the contribution of the content-based approach that provides only two items at most, but has a significant impact in the results obtained by the hybrid approach. The main reason for this high impact is that the two items recommended by this approach are put in the top of the list, positions highly scored, and this items are based on the content of the last user activities.

In accordance with the results presented in the previous section, collaborative filtering is the approach with the highest impact in our hybrid recommender system. The names suggested by this approach are also in the top of the list (two positions after the content-based approach) and capture the diversity of names suggested by similar users, ensuring good results for the hybrid approach.

Although with a small contribution, the popularity-based approach has its importance. The popularity-based approach fills a great part of the recommendation list (the tail of the list) that would not be filled properly by any of the other approaches studied, ensuring that all recommended lists have one thousand names.

6 Final Remarks

The task of recommending given names is very hard. Even though the list of recommend names is quite long, with up to 1000 names, it is still difficult to predict given names of interest for many users, even with a hybrid system that combines efforts from different approaches. There are many cultural factors and behavior (or even fashion!) that may influence the choice of a name. A deeper understanding of such factors, particularly with a much larger and representative dataset, can be a fruitful track to build more efficient recommender systems for given names.

Acknowledgments. The authors would like to thank Ivo Romário Lima and Matheus Cardoso Silva for their collaboration during system development.

References

1. R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.
2. J. Liu, P. Dolan, and E. Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.
3. C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
4. F. Mitzlaff and G. Stumme. Namelings - discover given name relatedness based on data from the social web. In *Proceedings of the International Conference on Social Informatics (SocInfo)*, pages 531–534, 2012.
5. M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.
6. F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, chapter 1, pages 1–35. Springer, Boston, MA, 2011.
7. J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.
8. F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.
9. U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.
10. J. Zobel and P. Dart. Phonetic string matching: lessons from information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information retrieval*, pages 166–172, 1996.

Collaborative Filtering Ensemble for Personalized Name Recommendation

Bernat Coma-Puig*, Ernesto Diaz-Aviles, and Wolfgang Nejdl

L3S Research Center, Leibniz University Hannover, Germany
{coma-puig, diaz, nejdl}@L3S.de

Abstract Out of thousands of names to choose from, picking the right one for your child is a daunting task. In this work, our objective is to help parents making an informed decision while choosing a name for their baby. We follow a recommender system approach and combine, in an ensemble, the individual rankings produced by simple collaborative filtering algorithms in order to produce a personalized list of names that meets the individual parents' taste. Our experiments were conducted using real-world data collected from the query logs of *nameling* (`nameling.net`), an online portal for searching and exploring names, which corresponds to the dataset released in the context of the ECML PKDD Discover Challenge 2013. Our approach is intuitive, easy to implement, and features fast training and prediction steps.

Keywords: Top-N recommendation; personalized ranking; given name recommendation

1 Introduction

There are many considerations when parents are deciding on a name for their child. Many parents choose to name their babies after a grandparent, other relative, or a close friend. Some others pick names from the actors or actresses of their favorite soap opera. Cultural and societal rules, the meaning of the name, family's traditions, or religious beliefs also play an important role in many countries at the time of choosing a given name for a baby.

This is indeed a daunting task for the parents and their decision will mark the child for the rest of his or her life. The given name should be unique, making the bearer stand out from the crowd, but at the same time it should also avoid embarrassment of being the source for nicknames, humiliating initials, or annoying email addresses[1].

From thousands of names to choose from, how do parents pick the right one for their baby? In this paper, we present an approach to help parents dealing with this information overload problem. In particular, we take a recommender systems approach

* Work done at the L3S Research Center as part of the ERASMUS exchange program while a student at Universitat Politcnica de Catalunya – BarcelonaTech (UPC) <bernat.coma@est.fib.upc.edu>.

[1] such as the one of our friend *H. Thomas Eatons*, who has the (unfortunate) email address of eatonsht@<anonymized>.com :) .

and show how an ensemble of simple collaborative filtering algorithms can help users to find given names that match their needs from a big pool of names.

We conduct this study in the context of the ECML PKDD'13 Discovery Challenge. This paper documents the approach of team "cadejo" on the offline phase of the challenge.

The main contribution of this paper is an intuitive approach for the task of given name prediction that is easy to implement, and that features fast training and prediction steps. Our study shows that, in this particular task, our ensemble of simple collaborative filtering building blocks performs significantly better than state-of-the-art latent factor models.

1.1 Preliminaries

We consider the dataset as sparse matrix $\mathbf{X} = [x_{ui}]$, where we use through this paper the letter u for users and i for names, which corresponds to the items in a recommender system setting. We use bold letters for matrices and vectors, and non bold for scalars. The set of users and names[2] are denoted by $\mathcal{U}$ and $\mathcal{I}$, respectively. Predictions for user-item pairs are denoted as $\hat{x_{ui}}$. The set of names that the user has interacted with is written as $\mathcal{I}(u)$. The set of users, who interacted with name i is $\mathcal{U}(i)$.

We use the notation $C_u(i)$ to represent the set of names co-occurring with name i within $\mathcal{I}(u)$, that is, $C_u(i) := \{j \mid i, j \in \mathcal{I}(u) \wedge i \neq j\}$.

We denote the bag of co-occurring names for a given item i, as follows:

$$C(i) := \bigcup_{u \in \mathcal{U}} \{((i, j), m(i, j)) \mid i \in \mathcal{I}(u) \wedge j \in C_u(i)\} ,$$

where $m(i, j) : \mathcal{I} \times \mathcal{I} \rightarrow \mathbb{N}_{\geq 1}$ is a function from the set name pairs $(i, j) \in \mathcal{I} \times \mathcal{I}$ to the set $\mathbb{N}_{\geq 1}$ of positive natural numbers. For each pair of names (i, j), $m(i, j)$ represents the number of occurrences of such pair in the bag, i.e., its *multiplicity*. The aggregated bag C over all items corresponds to $C := \bigcup_{i \in \mathcal{I}} C(i)$.

We use S_u to represent the user u's sequence of interactions ordered according to the corresponding timestamp , e.g., if user u searches first for name i_1, after that for i_4 and finally for name i_2, then his sequence S_u is represented as:

$$S_u = i_1 \longrightarrow i_4 \longrightarrow i_2.$$

For example, consider three users u_1, u_2 and u_3, and their corresponding sequences S of search actions in temporal order:

$$S_{u1} = i_1 \longrightarrow i_4 \longrightarrow i_2 \longrightarrow i_3$$
$$S_{u2} = i_4 \longrightarrow i_5 \longrightarrow i_1 \longrightarrow i_4 \longrightarrow i_3$$
$$S_{u3} = i_3 \longrightarrow i_5 \longrightarrow i_6 \longrightarrow i_7 \longrightarrow i_4$$

[2] In this paper, we use the terms "names" and "items" interchangeably.

The bag of co-occurrences for item i_4, $C(i_4)$, sorted in decreasing order of multiplicity, is given by:

$$C(i_4) = \{((i_4, i_3), 3), ((i_4, i_1), 2), ((i_4, i_5), 2), ((i_4, i_2), 1), ((i_4, i_6), 1), ((i_4, i_7), 1)\} .$$

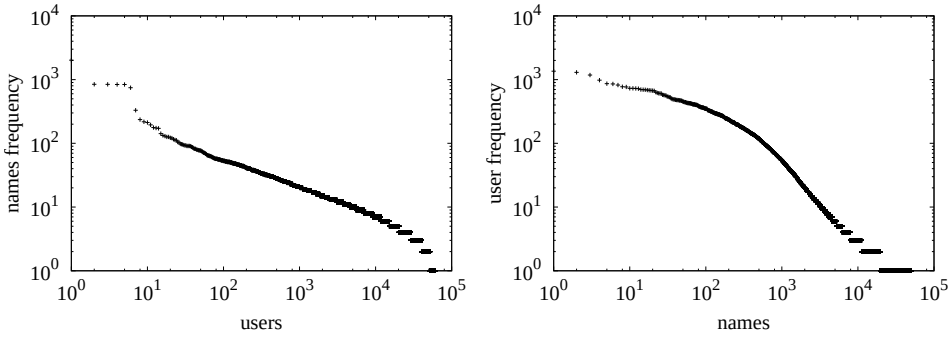
1.2 The Dataset

The dataset provided for the offline challenge is based on the query logs of *nameling* (`nameling.net`), an online portal for searching and exploring names. The collection comprises the time period from March 6th, 2012 to February 12th, 2013. In total the dataset contains 515,848 interactions (i.e., activities) from 60,922 different users and 50,277 unique names. Figure 1a shows the frequency of names per user. We can observe that it corresponds to a characteristic graph of a long-tail distribution, where few names tend to concentrate a large number of users. The frequency of users per given name is shown in Figure 1b.

There are 5 different types of interactions within the dataset, which are described as follows:

1. ENTER_SEARCH: the user explicitly writes a name in the search field of nameling's website in order to search for it.
2. LINK_SEARCH: the user clicks on a name of showed names at nameling's website, following a link to a search result page.
3. NAME_DETAILS: the user requests more detailed information of a name.
4. LINK_CATEGORY_SEARCH: Wherever available, a name is categorized according to the corresponding Wikipedia article. Users may click on such a category link to obtain all names in the corresponding category.
5. ADD_FAVORITE: the user saves the name in his list of favorite names.

In addition to these datasets there is a list of valid or *known names* provided by the organizers of the challenge, which contains 36,436 given names.



(a) **Frequency of names per given user.** (b) **Frequency of users per given name.**

Figure 1. Frequency of users and names.

1.3 The Task

The task for the offline challenge is to recommend a personalized ranked list of names for each user in the test set, based on the users' (partial) search history in nameling.

The recommender system's quality is evaluated with respect to the set of names that users have entered directly into nameling's search field. The rationale to restrict the evaluation to ENTER_SEARCH activities is that all other user activities are biased towards the lists of names which were displayed to nameling users.

The test set is built by taking from the training data the chronologically last two names, which had directly been entered into nameling's search field (i.e., using the ENTER_SEARCH activity) and which are also contained in the list of known names as detailed in the challenge description[3].

The assessment metric for the recommendations is Mean Average Precision at a cut-off of 1000 (MAP@1000) [1]. That is, for each test user look up the left-out names and take the precision at the respective position in the ordered list of recommended names. These precision values are first averaged per test user and than in total to yield the final score. MAP@1000 means that only the first 1,000 positions of a list are considered. Thus it might happen that for some test users one or both of the left out names do not occur in the list of recommendations. These cases will be handled as if the missing names were ranked at position 1001 and 1002 respectively.

1.4 Data Preprocessing and Validation Set

In our study we could not find a clear mechanism on how to exploit activities of type LINK_CATEGORY_SEARCH, and therefore we drop such interactions from the dataset. We also concentrate only on names which appear as part of at least one interaction and that were also present in the known names list. In total our experiments consider a total number of 260,236 user-name pair interactions, from $|\mathcal{U}| = 60,922$ different users and $|\mathcal{I}| = 17,467$ unique names. The mean of names per user is 4.35, the median is 3 names per user, with a minimum 1 and a maximum of 1670 names per user.

To evaluate our results we built a cross-validation dataset using the script provided by the organizers of the challenge. The script gives us a validation with 13,008 users and two target names. From these 13,008 users, 2,264 are not within the 4,140 users in the test set, which are the ones we are required to give recommendations. In order to have a more representative cross-validation dataset, for each of these 2,264 users we also selected, from the remaining transactions in the training set, the last 2 names the user interacted with. Note that in this case we ignored the additional constraints imposed by the script, e.g., the type of activity.

2 Related Work

Although top-N recommender systems have been studied in depth, the particular task of recommending given names is rather new. For example recent work by Mitzlaff et al. studies the relatedness of given names based on data from the social web [11].

[3] `http://www.kde.cs.uni-kassel.de/ws/dc13/faq/` .

This work shows the importance of co-occurrence networks for the recommendation task. Our approach also exploits name co-occurrences in the Name-to-Name algorithm introduced in Section 3.

The NameRank algorithm introduced in [12] adapts FolkRank [8] for name recommendation, showing promising results. The algorithm basically solves a personalized version of PageRank [4] per user in the system, over a graph of names, which does not scale gracefully to large-scale data. Our approach, on the other hand, is flexible enough to combine multiple predictors from simple collaborative filtering models, which makes it more attractive for big data scenarios.

3 Methods

Collaborative Filtering (CF) algorithms are best known for their use on e-commerce Web sites, Online Social Networks, or Web 2.0 video sharing platforms, where they use input about a user's interests to generate a (ranked) list of recommended items. In this section, we describe the collaborative filtering models which are used by our approach as well as the assembling strategy to compute the final prediction.

3.1 Name-to-Name Collaborative Filtering

This approach for name recommendation is based on the classic item-based collaborative filtering algorithm introduced by Amazon.com [10]. This algorithm matches each user's interaction with a name to a set of similar names, then combines those similar items into a recommendation list.

To determine the most similar match for a given name, the algorithm constructs a bag of co-occurring names across all user interactions in the collection. The rationale behind this algorithm is that there are many names that do not co-occur in any of the user's name transactions ($\mathcal{I}(u)$), and thus the approach is efficient in terms of processing time in memory, since there is no need to compute similarities over all possible pairs of names in the collection.

To compute the final recommendation list, the algorithm finds names similar to each of the ones in the user's set of names $\mathcal{I}(u)$, aggregates those co-occurring names, and then recommends the most popular or correlated names. This computation is very quick, depending only on the number of names the user has interacted with.

The Name-to-Name algorithm is summarized in Algorithm 1[4].

3.2 Neighborhood-based Collaborative Filtering

Neighborhood-based recommendation is a classic approach for Collaborative Filtering that still ranks among the most popular methods for this problem. These approaches are quite simple to describe and implement featuring important advantages such as the ability to explain a recommendation and to capture "local" relations in the data, which

[4] In Section 4, we explain the models used in our ensemble and provide more details about Algorithm 1's functions `getRandomName`() and `getRandomCoName`() in this context.

Algorithm 1 NAME-TO-NAME CF

Input:
Target user $u \in \mathcal{U}$. Recommendations will be computed for this user;
$\mathcal{I}(u) \subset \mathcal{I}$: set of names that the user has interacted with;
C : bag of co-occurring names;
$N \in \mathbb{N}$: size of the recommendation list;
max_iterations: maximum number of iterations.
Output: Recs(u): ranked list of recommendations for user u.

procedure GETRECOMMENDATIONS(u, $\mathcal{I}(u)$, C, N, max_iterations)
  Recs(u) $\leftarrow \emptyset$
  Recs(u) $\leftarrow$ NAME-TO-NAME(u, $\mathcal{I}(u)$, C, N, Recs(u), max_iterations)
  while |Recs(u)| $< N$ **do**
    Recs(u) $\leftarrow$ NAME-TO-NAME(u, Recs(u), C, N, Recs(u), max_iterations)
  end while
  return `sort`(Recs(u))
end procedure

procedure NAME-TO-NAME(u, $\mathcal{I}'(u)$, C, N)
  while |Recs(u)| $< N$ **and** $t <$ max_iterations **do**
    $i \leftarrow$`getRandomName`($\mathcal{I}'(u)$)
    $((i, j), m) \leftarrow$`getRandomCoName`($C(i)$)
    if $j \notin$ Recs(u) **and** $j \notin \mathcal{I}'(u)$ **then**
      Recs(u) $\leftarrow$ Recs(u) $\cup \{(j, m)\}$
    end if
    $t \leftarrow t + 1$
  end while
  return Recs(u)
end procedure

are useful in making serendipitous recommendations. In particular, we used the Top-N variants of the User-Based and Item-Based algorithms [5] as part of our name recommendation ensemble.

3.3 Ensemble

Our solution to the challenge consists of an ensemble of individual rank estimates of a set of collaborative filtering algorithms, a method that has shown to improve the quality of the recommendations [3].

Since the value estimates of our models, $\hat{x}$, can be in different scales, we do not combine their values directly, but rather we use their rank estimates. Formally, the ensemble of the rank estimates of l models is given by:

$$\hat{x}_{ui}^{\text{rank}} := \sum_{l} \alpha_l \cdot \frac{1}{\text{rank}(\hat{x}_{ui}^{l})} \ , \tag{1}$$

where α_l is a weight associated to the predictors of model l, $\text{rank}(\hat{x}^l_{ui})$ is the rank position within the lth ranked list corresponding to the estimate value $\hat{x}^l_{ui}$. That is, the combined estimate $\hat{x}^{\text{rank}}_{ui}$ corresponds to the weighted reciprocal rank of the individual models.

4 Results

In this section, we detail the collaborative filtering models, report their parameters, and individual recommendation performance in terms of MAP@1000. We also present the performance boost achieved by our ensemble.

The ensemble of our solution consists of 9 collaborative models that we describe as follows.

[m0 – N2N-Freq] is a Name-to-Name CF model that is created using the names co-occurring with the names of a given test user, according to Algorithm 1. This model considers the "ENTER_SEARCH", "LINK_SEARCH" and "NAME_DETAILS" activites to build the bag of co-occurrences. We randomly select a name i for given test user u via the `getRandomName(·)` procedure specified in Algorithm 1, where the chance for a name to be chosen is proportional to how often user u has interacted with it, which adds a positive bias towards those names that are more searched by the user. Furthermore, we also bias the selection of the co-occurring name j (`getRandomCoName(·)` procedure in Algorithm 1) towards the multiplicity of the pair (i, j).

Example. To illustrate this approach, consider the following example. Our dataset consists of five users, $u_1 \dots u_5$, and our task is to predict a recommendation list of names for user u_1. The sequence of interactions for user u_1 is denoted as S_{u1} (cf. Section 1.1) is given by

$$S_{u1} = i_4 \longrightarrow i_1 \longrightarrow i_4.$$

and for the other four users, their corresponding sequences are:

$$S_{u2} = i_1 \longrightarrow i_4 \longrightarrow i_3$$

$$S_{u3} = i_4 \longrightarrow i_5 \longrightarrow i_1 \longrightarrow i_4 \longrightarrow i_3$$

$$S_{u4} = i_3 \longrightarrow i_6 \longrightarrow i_7 \longrightarrow i_4$$

$$S_{u5} = i_1 \longrightarrow i_5 \longrightarrow i_2$$

then, the bags of co-occurrences for the names in S_{u1}, i.e., i_4 and i_1, sorted in decreasing order of multiplicity, are given by:

$$C(i_4) = \{((i_4, i_1), 3), ((i_4, i_3), 3), ((i_4, i_5), 1), ((i_4, i_6), 1), ((i_4, i_7), 1)\}.$$

$$C(i_1) = \{((i_1, i_4), 3), ((i_1, i_3), 2), ((i_1, i_5), 2), ((i_1, i_2), 1)\}.$$

Using the N2N-Freq shown in Algorithm 1, we first chose one name from user u_1's names (i.e., from $\mathcal{I}(u_1)$), and the name's corresponding bag of co-occurrences. Let us assume that $i_4 \in \mathcal{I}(u_1)$ and its respective bag $C(i_4)$ are chosen.

The first item to be included in the list of recommendations is i_3 (i_1 would not be chosen because $i_1 \in \mathcal{I}(u_1)$).
In the next iteration, consider that $C(i_4)$ is selected again, given that it has a higher probability to be picked due to the frequency of item i_4 in the sequence S_{u1}. In this case, i_5 would be the item chosen to be included in the list of recommendations.
In the third iteration, the list selected is $C(i_1)$, then the first item to be selected is i_2. Note that there are no more items from $C(i_1)$ that can be included in the list. Then, $R(u_1)$ is filled up using items from $C(i_4)$.
Finally, the list of recommendations for u_1 corresponds to:

$$R(u_1) = [i_3, i_5, i_2, i_6, i_7] \ .$$

[m1 – N2N-Freq-ES] follows the same approach as model m0, but the bag of co-occurring names used to compute the predictions considers only the "ENTER_SEARCH" activity to build the bag of co-occurrences.

[m2 – N2N-Time] is also a Name-to-Name CF model similar to m0, but with the difference that the names $\mathcal{I}(u)$ are not selected biased towards frequency of user interactions, but towards recency. That is, the names included in the recommendation list are those that co-occur with the last searches of the test user. The goal of this model is to capture the latest user preferences as input to compute the recommendations.
Example. Using this algorithm, with S_{u1}, S_{u2}, S_{u3}, S_{u4} S_{u5}, $C(i_4)$ and $C(i_1)$ from the example given for **m0**. Using this algorithm, biased towards recency, all selectable items from $C(i_4)$ have a higher probability of being chosen. The firsts items would correspond to i_3, i_5 and i_6. From $C(i_1)$ the selectable items are i_7 and i_2. A possible recommendation list corresponds to:

$$R(u_1) = [i_3, i_5, i_6, i_7, i_2] \ .$$

[m3 – N2N-Time-ES] follows the same temporal strategy as m2, but the bag of co-occurring names only considers the "ENTER_SEARCH" activity.

[m4 – N2N-Time-NoTop5] this model is the same one as m2, but only the top-5 most popular names are excluded from the bag of co-occurrences. The rationale behind this model is to get a more specific list of names, avoiding the names that are too popular.

[m5 – N2N-Time-NoTop10] This model is similar to model m2, with the exception that the top-10 most popular names in the collection have not been considered to build the bag of co-occurrences.

[m6 – UB-T] is a user-based collaborative filtering algorithm [5] using Tanimoto coefficient for binary feedback as similarity metric [14]. We used a neighborhood of size 100[5].

[5] Observe that we did not optimize for this parameter.

Model	Description	MAP@1000
m0	N2N-Freq	0.033449
m1	N2N-Freq-ES	0.033430
m2	N2N-Time	0.032296
m3	N2N-Time-ES	0.032008
m4	N2N-Time-NoTop5	0.032526
m5	N2N-Time-NoTop10	0.032455
m6	UB-T	0.023921
m7	UB-LL	0.028365
m8	PR	0.026483
	Final ensemble	**0.036766**
baseline	Most Popular Names	0.028138

Table 1. Recommendation performance in terms of MAP@1000 for the individual models and the final ensemble. The performance of a non-personalized model that always recommends the most popular 1000 names is reported as baseline.

[m7 – UB-LL] is a user-based model that uses likelihood as similarity metric. As in the previous model, we also used a neighborhood of size 100 in this case.

[m8 – PR] This model corresponds to PageRank [4] computed on the graph of co-occurring names. This is a non-personalized recommendation algorithm biased to the most popular items. We used this algorithm to "fill up" recommendation lists with less than 1000 names per user.

All models, except m6 and m7, were implemented in the Python programming language, using the numeric libraries of NumPy and SciPy[6]. For the user-based models (m6 and m7), we used the Java implementation provided by Apache Mahout[7].

Table 1 summarizes the individual performance of these models. We also report the performance of a non-personalized model that always recommends the most popular 1000 names.

Engineering the Final Ensemble

We compute the final ensemble by first combining different *flavors* of the same approach, and then combining the resulting ranked lists as explained in Section 3.3. Figure 2 illustrates the assembling process.

All weights (the α's in Equation 2) were determined experimentally based on the performance achieved by the (sub-)ensambles in our cross validation set.

[6] `http://www.scipy.org/` .
[7] `http://mahout.apache.org/` .

We found that the best way to combine the N2N-Freq* (m0 and m1) and N2N-Time* (m2 and m3) algorithms was by giving them equal weights, this is not surprising given their very similar performance. On the other hand, the performance of the User Based algorithms differs more substantially. In this case, we found that the best way to combine them was by giving a higher weight to UB-LL ($\alpha_{UB-LL} = 0.8$) and a weight of $\alpha_{UB-T} = 0.2$ to UB-T, for a UB combination (m6 + m7) that achieved a MAP@1000 of 0.028880.
We combine the unpersonalized ranked list output by the PageRank (m8) with the UB ensemble to fill up user's lists with less than 1000 items, using an asymmetric weighting scheme, favoring the UB combination.

The final ensemble combines the N2N family combinations with the ranked list from the filled UB models. We found that the best combination was obtained by giving the N2N and UB*+PR a weights of 0.8 and 0.2, respectively. The MAP@1000 for the final ensemble reaches a value of 0.036766. Please refer to Table 1 to compare the ensemble's performance to the one of the individual models.

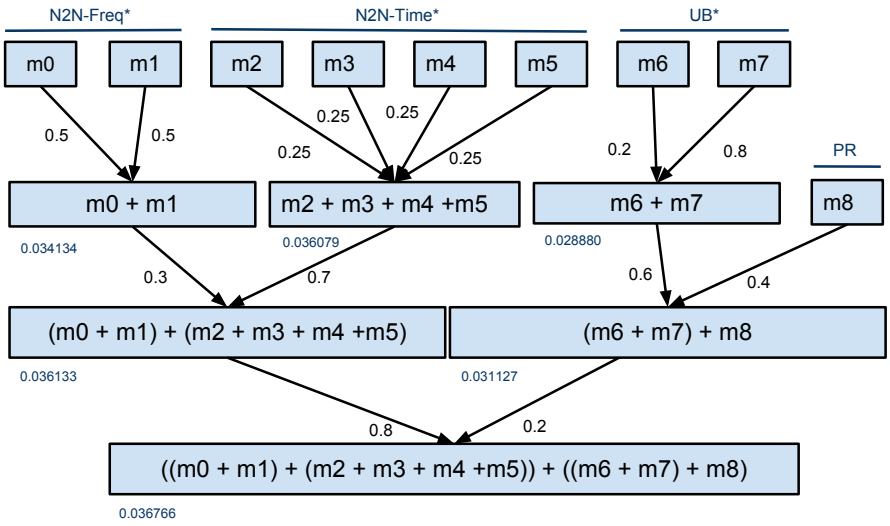


Figure 2. Final ensemble. The α weights for the partial model ensembles are indicated next to the corresponding arrow. The symbol '+' indicates the assembling of the models. The MAP@1000 for the corresponding sub-ensembles are shown below the respective boxes.

5 General Thoughts

The low values of MAP@1000 obtained by our approach on this dataset give an idea of how difficult the problem of recommending given names is.

Given the evaluation design of hiding the last two names the user interacted with, models that capture the latest user preferences, e.g., from the session information, tend to work well for us.

Neighborhood-based algorithms perform worse than item-to-item co-occurrences. Within the item-based and user-based variants, we observed that results from item-based collaborative filtering were inferior to the ones achieved by the user-based models, and therefore we did not consider them in the ensemble.

One of CF's most successful techniques are low dimensional linear factor models, that assume user preferences can be modeled by only a small number of latent factors. One of such methods is matrix factorization, which has been effectively used for the rating and item prediction task [9].

We conducted extensive experiments using state-of-the-art CF algorithms based on matrix factorization. In particular, we evaluated the performance of BPR [13] and RMFX [6] for the challenge's task, but we found that the performance achieved was only at the level of a baseline predictor that recommends the most popular names. This poor performance of matrix factorization models has been also observed by Folke et al. [12].

We also learned a name-to-name similarity matrix from the co-occurring names adjacency via optimizing a ranking criteria, as suggested in [13], the results were also discouraging.

Furthermore, we also tried to optimize directly for MAP following a Learning to Rank framework suggested in [7] and [2]. This approach learns the latent factors for users and items, and then applies standard Learning to Rank to optimize for a desired metric. Our results did not reach the level of the baseline predictor of most popular names.

Given this performance, we did not include any latent factor model in our ensemble. Why the results achieved using latent factor models, for this particular task of name prediction, are inferior to the ones obtained with simple methods? In our future research, we plan to explore this question more in detail.

6 Conclusion

In this paper, we presented an ensemble of several algorithm for personalized ranked recommendation of given names. We found that the co-occurring name information was a key component for the Name-to-Name algorithms used in our ensemble. Our method is intuitive and simple to implement, and does not suffer from the scalability issues as previous methods introduced for this task.

As a future work, we plan to further explore this interesting challenge in order to help parents deciding what is the best name for their baby.

Acknowledgements

We would like to thank Asmelash Teka and Rakshit Gautam for their valuable feedback. This work is funded, in part, by the L3S IAI research grant for the *FizzStream!* Project. Bernat Coma-Puig is sponsored by the EuRopean Community Action Scheme for the Mobility of University Students (ERASMUS).

References

1. R. Baeza-Yates, G. Navarro, and N. Ziviani. *Modern Information Retrieval*. Addison-Wesley, 2nd edition, 2011.
2. S. Balakrishnan and S. Chopra. Collaborative ranking. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 143–152, New York, NY, USA, 2012. ACM.
3. R. M. Bell, Y. Koren, and C. Volinsky. The bellkor solution to the netflix prize. Technical report, AT&T Labs, 2007.
4. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.
5. C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 107–144. Springer, 2011.
6. E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. Real-time top-n recommendation in social streams. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 59–66, New York, NY, USA, 2012. ACM.
7. E. Diaz-Aviles, M. Georgescu, and W. Nejdl. Swarming to rank for recommender systems. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 229–232, New York, NY, USA, 2012. ACM.
8. A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In Y. Sure and J. Domingue, editors, *ESWC*, volume 4011 of *Lecture Notes in Computer Science*, pages 411–426. Springer, 2006.
9. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.
10. G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.
11. F. Mitzlaff and G. Stumme. Onomastics 2.0 - the power of social co-occurrences, 2013.
12. F. Mitzlaff and G. Stumme. Recommending given names, 2013.
13. S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.
14. T. Tanimoto. IBM internal report 17th nov., 1957.

Nameling Discovery Challenge - Collaborative Neighborhoods

Dirk Schäfer and Robin Senge

Mathematics and Computer Science Department
Philipps-Universität Marburg, Germany
dirkschaefer@jivas.de, senge@informatik.uni-marburg.de

Abstract. This paper describes a series of experiments designed to solve the "Nameling" challenge. In this task, a recommender should provide suggestions for interesting first names, based on a set of names in which a user has shown interest. An approach based on dyadic factors is proposed where side-information about names and users were incorporated. Furthermore, factors based on User-based Collaborative Filtering play a central role. The performance considering the neighborhood and binary similarity measures was assessed.

Keywords: collaborative filtering, implicit feedback, dyad, competition

1 Introduction

Implicit feedback data can be collected whenever users are interacting with information systems. In connection with recommender systems this data is appealing, because it is obtainable in large quantities, e.g. from log files, and can be used as a complementary information source to rating data. On the one hand, the popularity of this kind of data is reflected by the numerous synonyms[1] in literature [6, 5, 12, 11]. One the other hand, there are various applications ranging from basket case analysis [7] to large scale news recommendation [2] and applied machine learning fields, e.g. Information Retrieval and Recommender Systems research to name a few.

2 The Challenge

The Nameling discovery challenge is part of a workshop held at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases[2] 2013.

[1] 0-1 data, one-class data, positive-only feedback, click-stream data, market basket data, click-through data
[2] ECML PKDD

2.1 Task Description

Given is a set of names that has been recorded as input by users of the Nameling web platform [8]. The task consists in recommending further names for a subset of selected users. The recommender should build an ordered list of one thousand names for each test user, where the most relevant names are placed at high rank positions. As performance metric the Mean Average Precision (MAP@1000) had to be used.

Table 1. Notations used in the paper

Symbols	Definition
$\mathcal{U}$	Set of all users
$\mathcal{I}$	Set of all items
u,v	Indices for users
i,j	Indices for items (=names)
$\mathcal{I}$(u)	Item set of user u
$\mathcal{U}$(i)	Set of users that have an affiliation with item i
w_{uv}	Similarity between two item sets
s_{uj}	Propensity for user u to select item j
$\mathcal{N}$	Items that are listed in the file "Namelist.txt"

2.2 Dataset

The dataset contains a training set which is a sparse matrix consisting of 59764 users (rows) and 17479 names (columns) and a test set with 4140 user IDs. Furthermore, a namelist file[3] is provided consisting of 44k names. For each user of the test set, two names from the system activity "ENTER_SEARCH", that are also contained in the namelist, have been extracted and are held out by the challenge organizers. Two Perl scripts were provided, the first one is able to build an own training and test set with names from the challenge training set exactly the same way as the challenge training and test set were built. The second script can be used for evaluation. In Figure 1 the sparsity of the training set is reflected by the frequencies of the most often chosen names. It shows a long-tail distribution, i.e. a small amount of names is very popular and at position 2000 names occur, which were only chosen by few users.

3 Related Work

In recommender systems literature, algorithms are classified as either being neighborhood-based (aka memory-based) or model-based. In the first group, there are user-based and item-based collaborative filtering methods. For user-based k-nearest neighbor collaborative filtering (UCF) the idea is to identify a

[3] Namelist.txt, see abbreviation $\mathcal{N}$ in Table 1

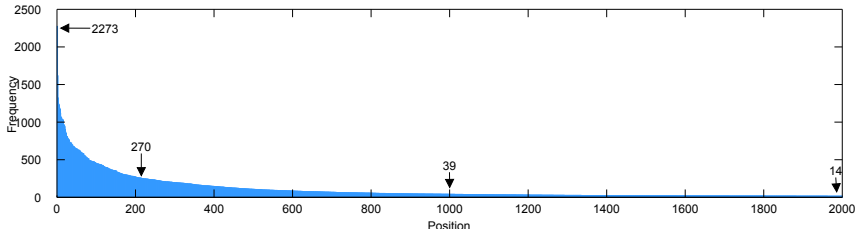


Fig. 1. Frequencies of top 2000 names.

group of k most similar users for each user to infer a ranking of items that are new for the user and also most interesting. Item-based nearest neighbor CF in contrast identify new items that have a relation to the existing item set of a user. The item-based CF approach has the advantage that a recommendation is easily explainable to the user and much more efficient to generate compared to UCF. On the other hand it lacks in accuracy.

For the model-based approaches and especially for this setting of implicit feedback, various methods based on Matrix Factorization (MF) have been created. In the One-Class Collaborative Filtering framework from Pan et al. two extreme assumptions about the data are being made [11], these are, that all missing values are all negative (AMAN) and all missing values are unknown (AMAU). They use weighted matrix factorization and sampling strategies to cope with the uncertainty about the unobserved data.

Another MF method that deals with side-information has been proposed in [4], where an embedding of auxiliary information into a weighted MF has been proposed. In [12] the Bayesian Personalized Ranking framework for implicit feedback has been described where also MF in combination with a bootstrap sampling approach is used to learn from pairwise comparisons.

NameRank is an item-based recommendation approach that has been proposed recently in combination with the Nameling data set and showed very good performance compared to the above mentioned approaches [10].

4 Recommendations Using Various Dyadic Factors

To begin with, we describe how dyadic factors can be used to induce rankings on names. The rest of the section deals with the engineering of these factors with two different approaches. The first one aims at improving the most popular items approach by using side-information for users and names, whereas the second approach is based on Collaborative Filtering.

4.1 Basic Scoring Scheme

In the following, each user-item pair (a dyad) receives a score based on the following equation:

$$s_{ui} = d_{ui} \cdot f_{u,i}^{[1]} \cdot f_i^{[2]} \tag{1}$$

Sorting s_{ui} scores in descending order for user u provides a ranking of items that can be recommended. The formula consists of a dyadic factor d_{ui} and two filter factors. The filters are indicator functions that adress the requirements of the prediction task. $f^{[1]}$ is 0 for names that are members of the user item-set $I(u)$. And the purpose of the other filter $f^{[2]}$ is to comply with the demand to accept only names that are part of the namelist $\mathcal{N}$. The various possibilites to engineer the dyadic factor are described below.

4.2 Discovering the "Most Popular" Baseline

In first experiments we found out that by simply considering a constant top 1000 majority vote for all names[4], we were already able to outperform some of the results others contributed to the leaderboard at an early stage. We could even improve this result by considering the 2000 top items and filtering out the training names of the individual users, which lead to the definition of the filter factor $f^{[1]}$. It turned out, that recommending items that way is known in literature as the Most Popular (MP) approach [10]. In the course of the challenge, we could identify other teams that scored equally, and we think they recommended names using the same approach. Because of this and its simplicity we say we discovered the "baseline method" within the leaderboards.

4.3 Side-Information on Names

In the given data set, there were no additional attributes on the item objects provided. Therefore we constructed two features as side-information for names: length of name and gender. Both features are characterized by having a finite set of discrete values as co-domain. With that, the now explained general procedure is applied, where the Most Popular Ranking is used as input. The idea is to split the MP Ranking into several queues corresponding to the available discrete values of a feature and later to adjust the MP recommendations for some of the users on basis of their item sets. From the queues, items are sampled according to proportions found in the item sets of the user. Since the item sets are typically small (see Figure 2), some significance criteria have to be met, e.g. a minumum size of the item-sets. Having found a ranking of at least 1000 names that way, we turned the ordered list of names in to numerical factors by assigning score values uniformly according to the rank position in an interval [max, min], e.g. the top ranked names recieved scores of $1, 0.9, 0.8, \ldots$[5].

Name Length Factor The general strategy of sampling from queues described above had to be slightly adapted due to the fact that there are 13 discrete values for name lengths, but item sets of users are too small to capture the proportions sufficiently (see Figure 2). To be able to sample from all those queues, the user

[4] that were used by the approx. 60k training users
[5] we actually used the interval [1,0.1]

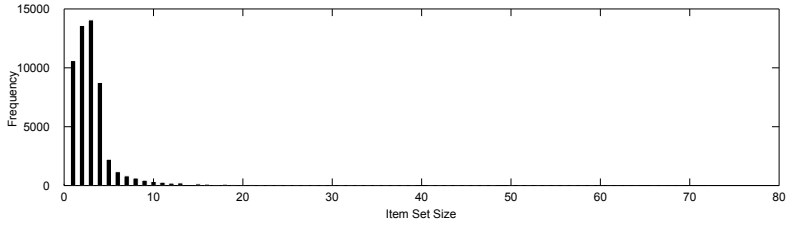


Fig. 2. Item set sizes across the training data

preferences for items, in this case for short or long names, must be significantly explainable. For this reason we decided to join the discrete values in two almost equally balanced sets: the set of shorter names (length 1 to 5) and longer names (length 6 to 13), see Table 2.

Table 2. Frequencies according to different lengths of names

Length of Name	1	2	3	4	5	6	7	8	9	10	11	12	13
Frequency	7	6	98	354	522	455	277	158	85	25	8	3	1

Gender Factor Since it was officially allowed and explicitly encouraged to use additional data sources for the offline challenge, we decided to include gender information for the names. By doing this, the Most Popular performance could be considerably improved (see Table 5). We extracted all names from the training set which were associated with the activity "ENTER_SEARCH" and used an external program[6] to classify names into the following three classes: 0 - unisex name or not identifiable, 1 - male, 2 - female. We used the idea described above and sampled from three queues according to the item sets' gender distributions until the new recommendation set reached a size of 1000 names.

4.4 Side-Information on Users

In previous uses of side-information, **one** global Most Popular recommendation was taken as basis and according to item set statistics the recommendation was modified by reordering or filtering. In further experiments, we followed a different approach: for groups of users, specified by their attributes, **many** Most Popular recommendation lists are created. Approximate geo locations of users were provided that were derived from IP addresses. We used in particular the country information for defining a new factor as described next.

[6] gender.c by Jörg Michael, which has been cited in the German computer magazine c't 2007: Anredebestimmung anhand des Vornamens.

Demographic Factor Additional geographical information was available for a subset of users. For every country we created a separate Most Popular recommendation list (see Table 3 for an excerpt). Unfortunately, the performance could only be increased marginally by this effort (see Table 5).

Table 3. The most popular names for seven countries are listed. The ? symbol represents the group of users that could not be geo-located.

Country	Frequency	Names						
?	36639	emma	anna	julia	michael	paul	thomas	christian
DE	19411	julia	emma	anna	michael	greta	emil	alexander
AT	1714	emma	paul	andreas	michael	anna	alexander	katharina
CH	661	max	sandra	christian	anna	daniel	katharina	jan
US	350	laura	sophie	emma	august	sarah	johann	leo
GB	134	matilda	emma	pia	martin	caroline	anja	robert
FR	99	sebastian	alma	simon	solveig	emma	lisa	emil
IT	79	emma	astrid	katharina	sophie	johanna	ida	verena

4.5 The User-based Collaborative Filtering Factor

User-based collaborative filtering is known as a successful technique to recommend items based on the **ratings** of items by different users. For this technique, the choice of a similarity measure that captures how similar users are, and the choice of neighborhood are the most important factors. In order to cope with this kind of binary data, the similarity has to be chosen suitably for implicit feedback data. And, as will be shown later, also the neighborhood size is even more crucial for the performance. The general user-based CF formula provides a score for each user-item pair as follows:

$$p_{uj} = \kappa \sum_{v=1}^{|\mathcal{U}|} w_{uv} c_{vj} \tag{2}$$

with indicator function

$$c_{ui} = \begin{cases} 1, & \text{if } i \in \mathcal{I}(u) \\ 0, & \text{else} \end{cases}$$

and a normalization term

$$\kappa = \frac{1}{\sum_{v=1}^{|\mathcal{U}|} w_{uv}}$$

to ensure that $p_{uj} \in [0, 1]$. Note that Equation (2) shows a special case, where the similarities of user u to all other users v are considered. In literature often only a neighborhood around u of the top N similar users are taken into account.

Similarity Measures A classical similarity measure between two vectors consisting of binary numbers is the Jaccard Index, which is the proportion between the sizes of the intersection and the union of two sets.

$$w_{uv} = \frac{|\mathcal{I}(u) \cap \mathcal{I}(v)|}{|\mathcal{I}(u) \cup \mathcal{I}(v)|} \tag{3}$$

In this context, it means two item-sets are more similar the more common items they share.

Good and often superior results were reported regarding the "log-likelihood similarity" which is implemented in the Mahout software package[7] for item- and user-based collaborative filtering [10, 3]. In computational linguistics Dunning proposed the use of the likelihood ratio test to find rare events as alternative to traditional contingency table methods [3]. In a bigram study he aimed at finding pairs of words that occurred next to each other significantly more often than expected from pure word frequencies. As basis for the log-likelihood statistics he used the following contingency table (see Table 4).

Table 4. Contingency table as used for the Dunning similarity. The first table shows the generic structure and below is the same table with the actual values in the UCF context.

	Event A	Everything but A
Event B	A and B together(k_{11})	B, but not A(k_{12})
Everything but B	A without B(k_{21})	Neither A nor B(k_{22})

	Event A	Everything but A
Event B	$\lvert\mathcal{I}(u) \cap \mathcal{I}(v)\rvert$	$\lvert\mathcal{I}(v)\rvert - \lvert\mathcal{I}(u) \cap \mathcal{I}(v)\rvert$
Everything but B	$\lvert\mathcal{I}(u)\rvert - \lvert\mathcal{I}(u) \cap \mathcal{I}(v)\rvert$	$\lvert\mathcal{I}\rvert - \lvert\mathcal{I}(u)\rvert - \lvert\mathcal{I}(v)\rvert + \lvert\mathcal{I}(u) \cap \mathcal{I}(v)\rvert$

The Dunning similarity, as we call it, is the log-likelihood ratio score defined[8] as follows:

$$w_{uv} = 2[H(k_{11} + k_{12}, k_{21} + k_{22}) + H(k_{11} + k_{21}, k_{12} + k_{22}) - H(k_{11}, k_{12}, k_{21}, k_{22})]$$

where $H(X)$ is the entropy defined as

$$H(X) = -\sum p(x) \log p(x).$$

Both similarity measures can be characterized regarding their use of information from two binary vectors under consideration. In [1] a survey of 76 binary similarity and distance measures had been carried out. All measures were described by a table called Operational Taxonomic Units that is identical to the contingency table shown above in Table 4. The measures fall in two groups regarding their use of negative matches that corresponds to the cell value k_{22}.

[7] http://mahout.apache.org/ - scalable machine learning libraries
[8] for implementation details refer to the Appendix

The Dunning similarity is not covered in that survey but falls into the category of similarity measures that make use of that kind of information. In contrast, the Jaccard similarity belongs to the negative match exclusive measures. The general inclusion or exclusion of negative matches has been debated over years and the particular choice of a measure is surely domain dependent.

4.6 Results

The MAP performances for the dyadic factor approaches based on MP are given in Table 5. Among those, the dyadic factor using user geo-locations performs best. Otherwise, the dyadic factor approaches based on UCF perform clearly better, if neighborhood sizes of UCF factor p_{ui} are larger than 500 (see Figure 3). The best performance values can be achieved when choosing the neighborhoods as large as possible. This means to consider all users according to Equation (2). To conclude, the choice between Dunning and Jaccard similarity for the UCF factor is not that relevant. In contrast, the neighborhood size is much more important. However, if not many users are available in the system and furthermore not much is known about users and items, the MP approach then provides a solid basis.

Table 5. MAP performance values for Most Popular experiments.

Method	Most Popular	Length of Name	Gender	Country
MAP	0.0247	0.0248	0.0252	0.0256

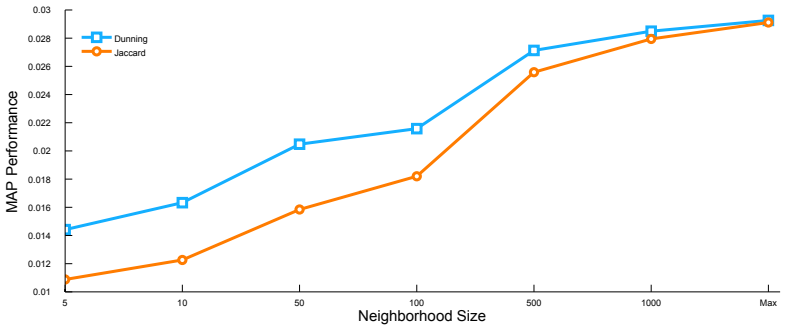


Fig. 3. Dyadic factor approach with different neighborhood sizes and two different similarity measures for the UCF factor. Mean MAP values measured on 50 test set splits of size 1000.

5 Hybridization by Combining Multiple Dyadic Factors

We propose to combine the different dyadic factors presented in the last section by extending the basic scheme. Furthermore, experimental results are shown for different factor combinations.

5.1 Extended Scoring Scheme

In this section, we show how different dyadic factors can be combined into a unified scoring scheme, which is an extension to Equation (1).

$$s_{ui} = D_{ui}^{\Theta} \cdot f_{u,i}^{[1]} f_i^{[2]} \tag{4}$$

$$D_{ui}^{\Theta} = (d_{ui}^{[1]})^{\gamma_1} \cdot (d_{ui}^{[2]})^{\gamma_2} \cdot \ldots \cdot (d_{ui}^{[m]})^{\gamma_m} \tag{5}$$

The choice of a dyadic factor combination is governed by the hyperparameter set $\Theta = \{\gamma_1, \gamma_2, \ldots\}$, where each parameter has the purpose of weighting the contribution of a particlar dyadic factor to the score s_{ui}.

5.2 Optimization

The optimization of the hyperparameter set Θ for formula (4) for $\forall u \in \mathcal{U}$ and $\forall i \in \mathcal{I}$ to maximize the overall MAP value is not trivial, because gradient based methods are not applicable here. For this reason we used grid search and an evolutionary algorithm[9] to find a well weighted combination of dyadic factors.

5.3 Experiments and Results

For a random selection of 1000 test users the following dyadic factors described in the last section were considered:

- The demographic factor c_{ui}, where the most popular items are recommended according to the country of a user.
- The length of a name factor n_{ui}.
- The gender of a name g_{ui}.
- User based Collaborative Filtering factor using Jaccard similarity p_{ui}.

According to Table 6 the combination of multiple dyadic factors can be beneficial. However, regarding the MAP performance only minimal improvements can be observed.

[9] CMA-ES from Apache Commons (http://commons.apache.org).

Table 6. MAP performance values for different combinations of factors

D_{ui}^{Θ}	Hyperparameters	MAP	Best Single Factor
$c^{\gamma 1} n^{\gamma 2}$	1.751, 0.066	0.0261	c_{ui} (0.0257)
$c^{\gamma 1} g^{\gamma 2}$	1.082, 0.941	0.0255	c_{ui} (0.0257)
$p^{\gamma 1} c^{\gamma 2}$	1.137, 0.516	0.0350	p_{ui} (0.0350)
$p^{\gamma 1} c^{\gamma 2} n^{\gamma 3} g^{\gamma 4}$	2.449, 1.365, 0.726, 0.877	0.0356	p_{ui} (0.0350)

6 Discussion

During the challenge phase we could confirm findings reported in literature: The baseline method described in section 4.2 performs well compared to statical methods using relational data from co-occurrence networks [9]. We found out that the choice of input for the recommendation has a large impact on the performance, e.g. restricting the item sets for the UCF approach a-priori to names contained in $\mathcal{N}$ has a negative effect. Furthermore, we introduced an approach based on weighted dyadic factors, which enabled us to combine different information sources and assumptions in a formal way. That allowed us to express the preferences of users by different factors and provides further possibilities, that are out of the scope of this paper, e.g. to combine User-based with Item-based Collaborative Filtering. Even though this approach seems to be appealing at first sight, the MAP performance improvements are only minimal compared to single dyadic factors. Introducing the various dyadic factors using side-information, they performed not as well as factors based on Collaborative Filtering, which shows the effectiveness of UCF despite its simplicity.

References

1. Seung-Seok Choi, Sung-Hyuk Cha, and C Tappert. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010.
2. Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.
3. Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74, 1993.
4. Yi Fang and Luo Si. Matrix co-factorization for recommendation with rich side information and implicit feedback. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pages 65–69. ACM, 2011.
5. Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 176–185. IEEE, 2010.

6. Michael Hahsler. Developing and testing top-n recommendation algorithms for 0-1 data using recommenderlab, 2010.
7. Andreas Mild and Thomas Reutterer. An improved collaborative filtering approach for predicting cross-category purchases based on binary market basket data. *Journal of Retailing and Consumer Services*, 10(3):123–133, 2003.
8. Folke Mitzlaff and Gerd Stumme. Namelings - discover given name relatedness based on data from the social web. In Karl Aberer, Andreas Flache, Wander Jager, Ling Liu, Jie Tang, and Christophe Guret, editors, *SocInfo*, volume 7710 of *Lecture Notes in Computer Science*, pages 531–534. Springer, 2012.
9. Folke Mitzlaff and Gerd Stumme. Relatedness of given names. *Human Journal*, 1(4):205–217, 2012.
10. Folke Mitzlaff and Gerd Stumme. Recommending given names. *arXiv preprint arXiv:1302.4412*, 2013.
11. Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE, 2008.
12. Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

Appendix

Calculation of the Entropies for the Dunning Similarity

The entropies for the Dunning similarities using the notation of Table 4 can be calculated as follows :

$$LLR = 2[H(row) + H(col) - H(row, col)]$$

$$H(row) = -((k_{11} + k_{12}) \log(k_{11} + k_{12}) + (k_{21} + k_{22}) \log(k_{21} + k_{22}))$$

$$H(col) = -((k_{11} + k_{21}) \log(k_{11} + k_{21}) + (k_{12} + k_{22}) \log(k_{12} + k_{22}))$$

$$H(row, col) = -((\sum_{ij} k_{ij} \log(\sum_{ij} k_{ij}) - (\sum_{ij} k_{ij} \log k_{ij})).$$

Improving the Recommendation of Given Names by Using Contextual Information

Marcos Aurélio Domingues[1], Ricardo Marcondes Marcacini[2], Solange Oliveira Rezende[1], and Gustavo E. A. P. A. Batista[1]

[1] Institute of Mathematics and Computer Science – University of São Paulo
Av. Trabalhador São-Carlense, 400, Cx. Postal 668, 13560-970, São Carlos, SP, Brazil
mad@icmc.usp.br, solange@icmc.usp.br, gbatista@icmc.usp.br
http://labic.icmc.usp.br
[2] Federal University of Mato Grosso do Sul
Três Lagoas, MS, Brazil
ricardo.marcacini@ufms.br

Abstract. The people who have to choose a given name, know how challenging it is to find a suitable name that fits the social context, the language, the cultural background, and especially, the personal taste. For example, future parents end up browsing through several lists of given names in order to choose a name for their unborn child. A recommender system can help a person in this task by recommending given names which are of interest to the user. In this paper, we exploit contextual information (e.g., time and location) in two state-of-the-art recommender systems for the task of recommending given names. The empirical results have shown that we can improve the recommendation of given names by using contextual information.

Key words: Given Names, Recommender Systems, Item-based Collaborative Filtering, Association Rules, Contextual Information

1 Introduction

The task of finding a suitable name for an unborn child is not so easy. Future parents usually face many books or web sites, listing given names, in order to find a suitable name for their child. Here, a suitable name is represented by a given name which satisfies a set of factors such as the social context, the language, the cultural background, and especially, the personal taste. Although this task is relevant in practice, little research has been performed on the analysis and application of interrelations among given names from a recommender system perspective. A recommender system is an information filtering technology which can be used to output an ordered list of items (e.g., given names) that are likely to be of interest to the user [1, 2].

In different scenarios, recommender systems are subject to scientific research, as, for example, recommending products [3], mobile applications [4], places of interest [5], movies [6], music [7]. In this paper, we exploit two state-of-the-art

recommender systems for the task of recommending given names. In addition, we also incorporate contextual information (e.g., time and location) in such systems in order to capture tendencies in choosing given names and, thus, to improve the recommendations.

The paper is organized as follows: In Section 2 we present some work related to the recommendation of given names. In Section 3 we describe the two state-of-the-art recommender systems used in this work and an approach to incorporate contextual information in these systems. We present our empirical evaluation in Section 4. We discuss the data set, the pre-processing of the data, the experimental setup and evaluation metric, and the empirical results. In Section 5 we show our contribution to the *15th Discovery Challenge: Recommending Given Names*. In Section 6 we present final remarks.

2 Related Work

In this section, we present some work related to the recommendation of given names. A search engine and recommender system for given name is described in [8]. This system, called *Nameling*[3], can be used by users to find a suitable given name. For example, the user enters a given name and obtains a browsable list of names.

In [9], the authors analyze the co-occurence of names from the *Nameling* system. They show how basic approaches from the field of social network analysis and information retrieval can be applied for discovering relations among names. In [10], a recommendation method for given names, based on co-occurence within Wikipedia[4], is proposed. The method, called preferential PageRank, is a modification of the well known PageRank algorithm [11]. There, the preferential PageRank method is evaluated by using a data set from the *Nameling* system.

An empirical evaluation comparing the preferential PageRank method against some state-of-the-art recommender systems, for the task of recommending given names, is presented in [12]. By using a data set from the *Nameling* system, the authors compare the user-based collaborative filtering [13], the item-based collaborative filtering [14], the weighted matrix factorization method [15], the most popular recommendation approach (that recommends the most popular names), and the random approach (that recommends names randomly) against the preferential PageRank method [10]. The results show that the preferential PageRank method provides good performance in terms of prediction accuracy as well as runtime complexity.

Our contribution for the *Nameling* system consists in exploiting the contextual information (i.e., the month and the city) contained in the data from the system to capture tendencies in choosing given names, and, thus, to improve the recommendations. Our proposal will be described in the next sections.

[3] `http://nameling.net/`
[4] `http://www.wikipedia.org/`

3 Recommender Systems

A recommender system for the web is an information filtering technology which can be used to predict preference ratings of items (e.g., movies, music, books, news, images, web pages, given names, etc) not currently rated by the user [16], and/or to output a set of items/recommendations that are likely to be of interest to the user [1, 2].

We focus our work on the task of selecting the top-N items/recommendations which are of interest to a user. We formalize this task as follows:

> Let p be the number of users $U = \{u_1, u_2, ..., u_p\}$ and q the number of all possible items that can be recommended $I = \{i_1, i_2, ..., i_q\}$. Now, let j be the number of historical sessions in a web site $S = \{s_1, s_2, ..., s_j\}$. Each session $s = \langle u, I_s \rangle$ is a tuple defined by a user $u \in U$ and a set of accessed items $I_s \subseteq I$. The set S is used to build a top-N recommendation model M.
> Given an active session s_a defined by an active user u_a and a set of observable items $O \subset I$, the recommendation model M uses the set O to identify the interest of the user u_a and recommend N items from the set of items/recommendations R, such that $R \subset I$ and $R \cap O = \varnothing$, that are believed to be the top preferences of the user u_a.

In this section, we present two state-of-the-art recommender systems: Item-based Collaborative Filtering and Association Rules based. In this work we use these systems for recommending given names. In addition, we present an approach which is used to incorporate contextual information in the two state-of-the-art systems in order to generate context-aware recommendations.

3.1 Item-Based Collaborative Filtering

The Item-based Collaborative Filtering technique analyzes items to identify relations among them [17]. Here, the recommendation model M is a matrix representing the similarities between all the pairs of items, according to a given similarity metric. An abstract representation of a similarity matrix is shown in Table 1. Each item $i \in I$ is an accessed item, for example, a given name.

Table 1. Item-item similarity matrix

	i_1	i_2	$\cdots$	i_q
i_1	1	$sim(i_1, i_2)$	$\cdots$	$sim(i_1, i_q)$
i_2	$sim(i_2, i_1)$	1	$\cdots$	$sim(i_2, i_q)$
$\cdots$	$\cdots$	$\cdots$	1	$\cdots$
i_q	$sim(i_q, i_1)$	$sim(i_q, i_2)$	$\cdots$	1

According to [17], the properties of the model and consequently the effectiveness of this recommendation algorithm depend on the method used to calculate

the similarity among the items. To calculate the similarity between pairs of items, for example, i_1 and i_2, we first isolate the users who have rated both of these items, and then, we apply a metric on the ratings to compute the similarity $sim(i_1, i_2)$ between i_1 and i_2. Metrics to measure the similarity between pairs of items are cosine angle, Pearson's correlation and adjusted cosine angle. In this paper, we use the cosine angle metric, defined as

$$sim(i_1, i_2) = cos(\overrightarrow{i_1}, \overrightarrow{i_2}) = \frac{\overrightarrow{i_1} . \overrightarrow{i_2}}{||\overrightarrow{i_1}|| * ||\overrightarrow{i_2}||}, \tag{1}$$

where $\overrightarrow{i_1}$ and $\overrightarrow{i_2}$ are rating vectors with as many positions as existing users in the set U. The operator "." denotes the dot-product of the two vectors. In our case, the rating vectors are binary. The value 1 means that the users accessed the respective item. The value 0 has the opposite meaning.

Once we obtain the recommendation model, we can generate the recommendations. Given an active session s_a containing a user u_a and its set of observable items $O \subseteq I$, the model generates the N recommendations as follows. First, we identify the set of candidate items for recommendation C by selecting from the model all items $i \notin O$. Then, for each candidate item $c \in C$, we calculate its similarity to the set O as

$$sim_{c,O} = \frac{\sum_{i \in K_c \cap O} sim(c, i)}{\sum_{i \in K_c} sim(c, i)}, \tag{2}$$

where K_c is a set with the k most similar items (the nearest neighbors) to the candidate item c.

Finally, we select the top-N candidate items with the highest similarity to the set O and recommend them to the user u_a.

3.2 Association Rules Based

A recommendation model M based on association rules is a set of rules. Each rule m has the form $m : X \rightarrow Y$, where $X \subseteq I$ and $Y \subseteq I$ are sets of items and $X \cap Y = \varnothing$. Here, we generate association rules with one single item in the consequent of the rule [18], i.e., Y is a singleton set. Each association rule is characterized by two metrics: support and confidence [19].

The support of a rule in a set of sessions S is defined as

$$support(X \rightarrow Y) = \frac{|X \cup Y|}{|S|}, \tag{3}$$

where $|X \cup Y|$ is the number of sessions in S that contain all items in $X \cup Y$ and $|S|$ is the number of sessions in S.

The confidence of a rule is the proportion of the number of sessions which contain $X \cup Y$ with respect to number of sessions that contain X, and can be formulated as

$$confidence(X \rightarrow Y) = \frac{|X \cup Y|}{|X|}. \quad (4)$$

Discovering all association rules from a set of sessions S consists in generating all rules whose support and confidence are greater than or equal to the corresponding minimal thresholds, called *minsup* and *minconf*. The classical algorithm for discovering association rules is Apriori [19].

To build the recommendation model M using association rules, the set of sessions S is used as input to an association rules algorithm to generate a set of rules. Once we have the model, we can make recommendations, R, to a new session. Given an active session s_a containing a user u_a and its set of observable items O, we build the set R as follows [18]:

$$R = \{consequent(m) | m \in M \text{ and } antecedent(m) \subseteq O \text{ and } consequent(m) \notin O\}. \quad (5)$$

To obtain the top-N recommendations, we select from R the N distinct recommendations corresponding to the rules with the highest confidence values.

3.3 The Weight Post-Filtering Approach

There are many definitions of context in the literature depending on the field of application and the available customer data [2]. For example, in [20], context is defined as any information that can be used to characterize an item. In this paper, we use time and location (i.e., month and city) as context to identify tendencies in the choice of a given name to improve the recommendations.

To incorporate contextual information in the previous recommender systems, we have extended the Weight Post-Filtering (PoF) approach, proposed in [21], for the task of item recommendation. The original approach was proposed for rating prediction [21].

The Weight PoF approach first ignores the contextual information in the data (in our case, the month and the city from each access) and applies a traditional algorithm (e.g., Item-based Collaborative Filtering or Association Rules based) to build the recommendation model. Then, it computes the probabilities of user's access items under a given context. The probability $P_c(u, i)$ that a user u accesses an item i under the context c can be computed as follows

$$P_c(u, i) = \frac{Num_c(u, i)}{Num(u, i)}, \quad (6)$$

where $Num_c(u, i)$ is the number of users that, like the user u, also accessed the item i under the context c; and $Num(u, i)$ is the total number of users that accessed the item i.

The score of the items computed by using the previous recommender systems are multiplied by the probabilities $P_c(u, i)$, incorporating context into the

recommendations and improving the performance of the recommender systems. Finally, the items are reordered and the top-N items are recommended to the user.

4 Empirical Evaluation

In this section, we empirically evaluate the recommender systems, presented in Section 3, in the task of recommending given names.

4.1 Data Set

The empirical evaluation is carried out using an usage data set from *Nameling*. According to [8], *Nameling* is a search engine and a recommender system for given names. In this system, the user enters a given name and obtains a browsable list of recommended names, called "*namelings*".

The data set is derived from the Nameling query logs, ranging from March 6th, 2012 to February 12th, 2013. It contains 515,848 accesses from 60,922 users to 20,714 different items (i.e., given names). There are five types of accesses/activities[5]:

1. **ENTER_SEARCH:** The user enters a name directly into the Nameling's search field;
2. **LINK_SEARCH:** The user follows a link on some result page;
3. **LINK_CATEGORY_SEARCH:** Wherever available, names are categorized according to the corresponding Wikipedia articles;
4. **NAME_DETAILS:** Users can get some detailed information for a name;
5. **ADD_FAVORITE:** Users can maintain a list of favorite names.

Additionally, for each access there are a timestamp and a proxy for the user's geographic location (i.e., country code, province, city, latitude and longitude) which is obtained by using the MaxMind's GeoLite City data base[6].

As part of the data set, there is also a list of known names[7] containing all names which are currently known in the *Nameling* web site. As we will see in Section 4.3, all names that occur in the evaluation data set are contained in this list of names.

4.2 Pre-processing of the Data Set

Before running the experiments, we pre-processed the data set by replacing invalid names and removing singleton sessions, as described below:

[5] We use the terms access and activity interchangeably.
[6] `http://www.maxmind.com/`
[7] `http://www.kde.cs.uni-kassel.de/nameling/dumps`

Replacing invalid names: In real-world data sets, it is common to find several variations of a name, for example, spelling variations due to typographical errors (like "Richard" and "Ricahrd") and differences in punctuation marks (like "O'Reilly" and "O Reilly"). Considering the existence of a reference list with valid names, we can use string comparison measures to replace an invalid name by the nearest valid name, thus believing that we are correcting a name typed incorrectly. Thus, in the data pre-processing step, we use the list of known names, described in the previous section, and apply the Jaro-Winkler measure [22] for detection and replacement of invalid names. For this purpose, it is necessary first to define the Jaro (j) measure between two strings w_1 e w_2 (Equation 7):

$$j(w_1, w_2) = \begin{cases} 0 & \text{if } h = 0 \\ \frac{1}{3}\left(\frac{h}{|w_1|} + \frac{h}{|w_2|} + \frac{h-t}{h}\right) & \text{otherwise,} \end{cases} \tag{7}$$

where h is the number of matching characters and t represents the number of transpositions. The matching between two characters c_1 and c_2, with $c_1 \in w_1$ and $c_2 \in w_2$ occurs when $c_1 = c_2$ and they are not further than $\frac{\max(|w_1|,|w_2|)}{2} - 1$. The number of transpositions is obtained by considering different orders for matching characters. The Jaro-Winkler (jw) is based on the Jaro measure, according to Equation 8, in which $l(w_1, w_2)$ represents the length of common prefix at the start of the string up to a maximum of 4 characters.

$$jw(w_1, w_2) = j(w_1, w_2) + \frac{l(w_1, w_2)}{4} * (1 - j(w_1, w_2)) . \tag{8}$$

The Jaro-Winkler measure was selected for this work because it shows better performance in studies involving name-matching [23].

Removing singleton sessions: For different reasons, users often access only one item on a web site and then leave it. The use of these sessions containing a singleton access by a recommender system can affect its accuracy negatively [24]. For example, singleton sessions will never count for the item-item similarity in the Item-based Collaborative Filtering technique. Thus, we have removed the singleton sessions from the data set.

After pre-processing the data, we obtained a set with 510,705 accesses from 55,779 users to 20,318 different names.

4.3 Experimental Setup and Evaluation Metric

To carry out the experiments, we use a Perl script[8] to split the data set in training and test sets.

The script selects some users with at least five different names for the test set. Then, for each test user, it withholds the last two entered names for evaluation.

[8] `http://www.kde.cs.uni-kassel.de/nameling/dumps/process_activitylog.pl`

To withhold the last two entered names, the script uses the following rules. For each test user, the script selects for evaluation the last two names which had directly been entered into the Nameling's search field (i.e., ENTER_SEARCH access) and which are also contained in the list of known names. The script only considers those names which were not previously added as a favorite name by the test user (i.e., ADD_FAVORITE access). Finally, the script removes the accesses after the names for evaluation and keeps in the test set only users with at least three accesses. The remaining users in the data set are used as training set.

To evaluate the recommender systems, we compute the metric Mean Average Precision (MAP) [25]. For each test user, the metric takes the left out evaluation names and compute the precision at the respective position in the ordered list of recommended names. These precision values are first averaged per test user and than in total to obtain the final score. Here, we use a Perl script[9] to compute the MAP@1000 which means that only the first 1,000 positions of a list of recommendations are considered.

With respect to the recommendation algorithms, we use the Item-based Collaborative Filtering and the Association Rules based, which were described in Section 3. In the Item-based Collaborative Filtering, the top-N recommendations are generated based on their 1, 5, 10, 15 and 20 most similar items (i.e., the 1, 5, 10, 15 and 20 nearest neighbors). In the Association Rules based algorithm, the recommendation models are built using a minimum support value determined to generate around 10,000; 50,000 and 100,000 rules. The minimum confidence values are defined as being the support value of the one thousandth most frequent item in the training set. This allows the generation of at least 1,000 rules without antecedent that can be used by default, in the case that no other rule applies. Here, as the left out names for evaluation are only names which had been directly entered into the Nameling's search field (i.e., ENTER_SEARCH access), we have selected only this type of access from the training set to build the recommendation models.

Finally, we use the month and the city from the accesses as contextual information in the Weight PoF approach, as described in Section 3.3. Such information can capture tendencies of names in a given city, in a given month. The month is obtained from the timestamp of the access. We obtain the city by using the proxy for the user's geographic location provided with the data set.

4.4 Empirical Results

We start by comparing the Item-based Collaborative Filtering technique (CF) against its contextual version that makes use of the Weight PoF approach (CF-PoF). In Table 2, we see that the values of MAP@1000 decrease when we increase the number of neighbors. This fact occurs because when we increase the number of neighbors, less similar items are used to generate the recommendations. Comparing the CF-PoF against the CF, we see an improvement of the

[9] `http://www.kde.cs.uni-kassel.de/nameling/dumps/evaluate_recommender.pl`

recommendations by using the contextual information. The CF-PoF algorithm provides gains of MAP@1000 ranging from 6.9% to 22.6%.

Table 2. Comparing the MAP@1000 values between CF and CF-PoF algorithms

K-neighbors	CF	CF-PoF
K = 1	0.0092	0.0099
K = 5	0.0038	0.0042
K = 10	0.0033	0.0039
K = 15	0.0031	0.0038
K = 20	0.0029	0.0031

In Table 3, we can see that the difference between the Association Rules based algorithm (AR) and its, respective, contextual version (AR-PoF) is quite small. In this case, the AR-PoF algorithm provides gains of MAP@1000 around 2.4%.

Table 3. Comparing the MAP@1000 values between AR and AR-PoF algorithms

Number of Rules	AR	AR-PoF
10,000	0.0337	0.0343
50,000	0.0314	0.0321
100,000	0.0290	0.0299

We also compare the results between both Tables 2 and 3. We see that the AR-PoF recommender system with 10,000 rules provides the best value for MAP@1000, i.e., 0.0343. If we compare this value against the one provided by the best recommender system in Table 2, the CF-PoF with K = 1, we see a gain of 246.5%. Besides, our Association Rules based algorithms are quite fast. We measured the computational time to build the recommendation model and generate the 1,000 recommendations. In our experimental scenario, we used an Intel Core i7 Ivy Bridge with a CPU clock rate of 3.4 GHZ, 32 GB of main memory, and running the Debian Linux operating system. To build a recommendation model, the algorithms take around 26 seconds (10,000 rules) to 2 minutes (100,000 rules). Here, the top-1000 recommendations are generated in approximately 1 second.

5 The 15th Discovery Challenge: Recommending Given Names

After analyzing the results presented in Section 4.4, we applied our best scenario to the data set from the *15th Discovery Challenge: Recommending Given Names*.

We pre-processed the data set, selected only names entered into the Nameling's search field, and then ran the algorithm which provided the best MAP@1000, i.e., the AR-PoF algorithm with about 10,000 association rules. With this scenario, our **Labic** team obtained a score of 0.0379 in the final leaderboard.

6 Final Remarks

Although the task of recommending given names is relevant in practice, little research has been performed on the perspective of recommender systems. In this paper, we exploited two state-of-the-art recommender systems in the task of recommending given names. In addition, we also incorporated contextual information in such systems to capture tendencies in choosing given names and, thus, to improve the recommendations. Although the gains obtained by using the Weight PoF approach are small, the results of our empirical evaluation present evidences that we can improve the recommendation of given names by using contextual information.

There are some directions to be explored in future research. For example, other pre-processing tasks can be applied on the data set in order to improve the quality of the data. We can also try other context-aware recommender systems in the task of recommending given names [26, 27]. On the other hand, we can also combine the two state-of-the-art recommenders, presented in this paper, in a hybrid algorithm.

Acknowledgments. This work was supported by the grants 2010/20564-8, 2011/19850-9, 2012/13830-9, 2012/07295-3, São Paulo Research Foundation (FAPESP).

References

1. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM **40**(3) (1997) 56–58
2. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B., eds.: Recommender Systems Handbook. Springer (2011)
3. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing **7**(1) (2003) 76–80
4. Karatzoglou, A., Baltrunas, L., Church, K., Böhmer, M.: Climbing the app wall: enabling mobile app discovery through context-aware recommendations. In: Proceedings of the 21st ACM international conference on Information and knowledge management. CIKM '12, New York, NY, USA, ACM (2012) 2527–2530
5. Baltrunas, L., Ludwig, B., Peer, S., Ricci, F.: Context-aware places of interest recommendations for mobile users. In Marcus, A., ed.: Design, User Experience, and Usability. Theory, Methods, Tools and Practice. Volume 6769 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 531–540
6. Ko, S.K., Choi, S.M., Eom, H.S., Cha, J.W., Cho, H., Kim, L., Han, Y.S.: A smart movie recommendation system. In Smith, M., Salvendy, G., eds.: Human Interface and the Management of Information. Interacting with Information. Volume 6771 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 558–566

7. Domingues, M.A., Gouyon, F., Jorge, A.M., Leal, J.P., Vinagre, J., Lemos, L., Sordo, M.: Combining usage and content in an online recommendation system for music in the long tail. International Journal of Multimedia Information Retrieval **2**(1) (2013) 3–13
8. Mitzlaff, F., Stumme, G.: Namelings: discover given name relatedness based on data from the social web. In: Proceedings of the 4th international conference on Social Informatics. SocInfo'12, Berlin, Heidelberg, Springer-Verlag (2012) 531–534
9. Mitzlaff, F., Stumme, G.: Onomastics 2.0 - the power of social co-occurrences (2013)
10. Mitzlaff, F., Stumme, G.: Relatedness of given names. Human Journal **1**(4) (2012) 205–217
11. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems **30**(1-7) (1998) 107–117
12. Mitzlaff, F., Stumme, G.: Recommending given names (2013)
13. Su, X., Khoshgoftaar, T.: A survey of collaborative filtering techniques. Advances in Artificial Intelligence (2009)
14. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW'01: Proceedings of the Tenth International Conference on World Wide Web, New York, NY, USA, ACM (2001) 285–295
15. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Eighth IEEE International Conference on Data Mining (ICDM'08). (2008) 263–272
16. Breese, J.S., Heckerman, D., Kadie, C.M.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. (1998) 43–52
17. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Transaction on Information System **22**(1) (2004) 143–177
18. Jorge, A.M., Alves, M.A., Azevedo, P.J.: Recommendation with association rules: A web mining application. In: Proceedings of Information Society (IS-2002): Data Mining and Warehouses, Ljubljana, Slovenia (2002)
19. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of Twentieth International Conference on Very Large Data Bases. (1994) 487–499
20. Dey, A.K.: Understanding and using context. Personal Ubiquitous Computing **5**(1) (2001) 4–7
21. Panniello, U., Gorgoglione, M.: Incorporating context into recommender systems: an empirical comparison of context-based approaches. Electronic Commerce Research **12**(1) (2012) 1–30
22. Winkler, W.E.: Methods for evaluating and creating data quality. Information Systems **29**(7) (2004) 531–550
23. Christen, P.: A comparison of personal name matching: Techniques and practical issues. In: Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops, Washington, DC, USA, IEEE Computer Society (2006) 290–294
24. Domingues, M.A., Soares, C., Jorge, A.M.: An empyrical study on the impact of singleton web accesses on the accuracy of recommender systems. In: Proceedings of the SBIA 2008 First Workshop on Web and Text Intelligence (WTI 08), Salvador, Bahia, Brazil (2008) 43–50
25. Voorhees, E., Harman, D., of Standards, N.I., (US), T.: TREC: Experiment and evaluation in information retrieval. Volume 63. MIT Press Cambridge (2005)

26. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems **23**(1) (2005) 103–145
27. Domingues, M.A., Jorge, A.M., Soares, C.: Dimensions as virtual items: Improving the predictive ability of top-n recommender systems. Information Processing & Management **49**(3) (2013) 698–720

Similarity-Weighted Association Rules for a Name Recommender System

Benjamin Letham

Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA, USA
bletham@mit.edu

Abstract. Association rules are a simple yet powerful tool for making item-based recommendations. As part of the ECML PKDD 2013 Discovery Challenge, we use association rules to form a name recommender system. We introduce a new measure of association rule confidence that incorporates user similarities, and show that this increases prediction performance. With no special feature engineering and no separate treatment of special cases, we produce one of the top-performing recommender systems in the discovery challenge.

Keywords: association rule, collaborative filtering, recommender system, ranking

1 Introduction

Association rules are a classic tool for making item-based recommendations. An association rule "$a \rightarrow b$" is a rule that item(set) a in the observation implies that item b is also in the observation. Association rules were originally developed for retail transaction databases, although the same idea can be applied to any setting where the observations are sets of items. As part of the ECML PKDD 2013 Discovery Challenge, in this paper we consider a setting where each observation is a set of names in which the user has expressed interest. We then form association rules "$a \rightarrow b$," meaning that interest in name a (or, in general, set of names a) implies interest in name b. The strength with which a implies b is called the *confidence* of the rule, and in Section 2.2 we explore different measures of confidence.

Association rules provide an excellent basis for a recommender system because they are scalable and interpretable. The scalability of association rule algorithms has been well studied, and is often linear in the number of items [1]. Using rules to make recommendations gives a natural interpretability: We recommend name b *because* the user has expressed interest in name a. Interpretability is an important quality of predictive models in many contexts, and is especially important in recommender systems, where it has been shown that providing the user an explanation for the recommendation increases acceptance and performance [2, 3].

One of the most successful tools for recommender systems, particularly at a large scale, is collaborative filtering [4, 5]. Collaborative filtering refers to a large class of methods, of which here we focus on user-based collaborative filtering and item-based collaborative filtering [6]. In user-based collaborative filtering, recommendations are made by finding the most similar users in the database and recommending their preferred items. In item-based collaborative filtering, similarity is measured between items and the items most similar to those already selected by the user are recommended. Like association rules, collaborative filtering algorithms generally have excellent scalability.

Our main contribution is to use ideas from collaborative filtering to create a new measure of association rule confidence, which we call *similarity-weighted adjusted confidence*. We maintain the excellent scalability and interpretability of collaborative filtering and association rules, yet see a significant increase in performance compared to either approach. Our method was developed in the context of creating a name recommender system for the ECML PKDD 2013 Discovery Challenge, and so we compare the similarity-weighted adjusted confidence to other collaborative filtering and association rule-based approaches on the Nameling dataset released for the challenge.

2 Similarity-Weighted Association Rule Confidence

We begin by introducing the notation that will be used throughout the rest of the paper. Then we discuss measures of confidence, introduce our similarity-weighted adjusted confidence, and discuss strategies for combining association rules into a recommender system.

2.1 Notation

We consider a database with m observations $x_1, \ldots, x_m$, and a collection of n items $Z = \{z_1, \ldots, z_n\}$. For instance, it may be m visitors to a name recommendation site, with Z the set of valid names. Each observation is a set of items: $x_i \subseteq Z, \forall i$. We denote the number of items in x_i as $|x_i|$.

We will consider rules "$a \rightarrow b$" where the left-hand side of the rule a is an itemset ($a \subseteq Z$) and the right-hand side is a single item ($b \in Z$). Notice that a might only contain a single item. We denote as $\mathcal{A}$ the collection of itemsets that we are willing to consider: $a \in \mathcal{A}$. One option for $\mathcal{A}$ is the collection of all itemsets, $\mathcal{A} = 2^Z$. If Z is very large this can be computationally prohibitively expensive and some restriction may be necessary. In our experiments in Section 3 we took $\mathcal{A} = Z$, that is, all itemsets of size 1.

2.2 Confidence and Similarity-Weighted Confidence

The standard definition of the confidence of the rule "$a \rightarrow b$" is exactly the empirical conditional probability of b given a:

$$\text{Conf}(a \rightarrow b) = \frac{\sum_{i=1}^{m} \mathbb{1}_{[a \subseteq x_i \textbf{ and } b \in x_i]}}{\sum_{i=1}^{m} \mathbb{1}_{[a \subseteq x_i]}}, \tag{1}$$

where we use $\mathbb{1}_{[\text{condition}]}$ to indicate 1 if the condition holds, and 0 otherwise.

This measure of confidence corresponds to the maximum likelihood estimate of a specific probability model, in which the observations are i.i.d. draws from a Bernoulli distribution which determines whether or not b is present. Because of the i.i.d. assumption, all observations in the database are considered equally when determining the likelihood that a implies b. In reality, preferences are often quite heterogeneous. If we are trying to determine whether or not a new user x_ℓ will select item b given that he or she has previously selected itemset a, then the users more similar to user x_ℓ are likely more informative. This leads to the *similarity-weighted confidence* for user x_ℓ:

$$\text{SimConf}(a \to b | x_\ell) = \frac{\sum_{i=1}^{m} \mathbb{1}_{[a \subseteq x_i \textbf{ and } b \in x_i]} \text{sim}(x_\ell, x_i)}{\sum_{i=1}^{m} \mathbb{1}_{[a \subseteq x_i]} \text{sim}(x_\ell, x_i)}, \tag{2}$$

where $\text{sim}(x_\ell, x_i)$ is a measure of the similarity between users x_ℓ and x_i. The similarity-weighted confidence reduces to the standard definition of confidence under the similarity measure $\text{sim}(x_\ell, x_i) = 1$, as well as

$$\text{sim}(x_\ell, x_i) = \begin{cases} 1, & \text{if } x_\ell \cap x_i \neq \varnothing. \\ 0, & \text{otherwise.} \end{cases}$$

Giving more weight to more similar users is precisely the idea behind user-based collaborative filtering. A variety of similarity measures have been developed for use in collaborative filtering, one of the more popular of which is the cosine similarity, which we use here:

$$\text{sim}(x_\ell, x_i) = \frac{|x_\ell \cap x_i|}{\sqrt{|x_\ell|}\sqrt{|x_i|}}. \tag{3}$$

2.3 Bayesian Shrinkage and the Adjusted Confidence

In [7], we show how the usual definition of confidence can be improved by adding in a beta prior distribution and using the maximum *a posteriori* estimate. The resulting measure is called the *adjusted confidence*:

$$\text{Conf}_K(a \to b) = \frac{\sum_{i=1}^{m} \mathbb{1}_{[a \subseteq x_i \textbf{ and } b \in x_i]}}{\sum_{i=1}^{m} \mathbb{1}_{[a \subseteq x_i]} + K}, \tag{4}$$

where K is a user-specified amount of adjustment, corresponding to a particular pseudocount in the usual Bayesian interpretation. In particular, the adjusted confidence is equivalent to there being an additional K observations containing a, none of which contain b. This reduces the confidence of "$a \to b$" by an amount inversely proportional to the support of a, allowing low-support-high-confidence rules to be used in the computation, but giving more weight to those with higher support. In terms of the bias-variance tradeoff, adjusted confidence leads to an increase in performance by reducing the variance of the estimate for itemsets

with low support. The Nameling dataset used here is quite sparse, so we add the same adjustment to our similarity-weighted confidence, producing the *similarity-weighted adjusted confidence*:

$$\text{SimConf}_K(a \to b|x_\ell) = \frac{\sum_{i=1}^m \mathbb{1}_{[a \subseteq x_i \textbf{ and } b \in x_i]} \text{sim}(x_\ell, x_i)}{\sum_{i=1}^m \mathbb{1}_{[a \subseteq x_i]} \text{sim}(x_\ell, x_i) + K}. \tag{5}$$

When $K = 0$, this reduces to the similarity-weighted confidence in (2).

2.4 Combining Association Rules to Form a Recommender System

The similarity-weighted adjusted confidence provides a powerful tool for determining the likelihood that $b \in x_\ell$ given that $a \subseteq x_\ell$. In general there will be many itemsets a satisfying $a \subseteq x_\ell$, so to use the association rules as the basis for a recommender system we must also have a strategy for combining confidence measures across multiple left-hand sides. For each left-hand side $a \in \mathcal{A}$ satisfying $a \subseteq x_\ell$, we can consider $\text{SimConf}_K(a \to b|x_\ell)$ to be an estimate of the probability of item b given itemset x_ℓ. There is a large corpus of literature on combining probability estimates [8, 9], from which one of the most common approaches is simply to compute their sum. Thus we score each item b as

$$\text{Score}(b|x_\ell) = \sum_{\substack{a \subseteq x_\ell \\ a \in \mathcal{A}}} \text{SimConf}_K(a \to b|x_\ell). \tag{6}$$

A ranked list of recommendations is then obtained by ranking items by score.

A natural extension to this combination strategy is to consider a weighted sum of confidence estimates. We consider this strategy in [10], where we use a supervised ranking framework and empirical risk minimization to choose the weights that give the best prediction performance. This approach requires choosing a smooth, preferably convex, loss function for the optimization problem. In [10] we use the exponential loss as a surrogate for area under the ROC curve (AUC), however in the experiments that follow in Section 3 the evaluation metric was mean average precision. Optimizing for AUC in general does not optimize for mean average precision [11], and we found that the exponential loss was a poor surrogate for mean average precision on the Nameling dataset.

2.5 Collaborative filtering baselines

We use two simple collaborative filtering algorithms as baselines in our experimental results in Section 3. For user-based collaborative filtering, we use the cosine similarity between two users in (3) to compute

$$\text{Score}_{\text{UCF}}(b|x_\ell) = \sum_{i=1}^m \mathbb{1}_{[b \in x_i]} \text{sim}(x_\ell, x_i) \tag{7}$$

For item-based collaborative filtering, for any item b we define $\text{Nbhd}(b)$ as the set of observations containing b: $\text{Nbhd}(b) = \{i : b \in x_i\}$. Then, the cosine similarity between two items is defined as before:

$$\text{sim}_{\text{item}}(b, d) = \frac{|\text{Nbhd}(b) \cap \text{Nbhd}(d)|}{\sqrt{|\text{Nbhd}(b)|}\sqrt{|\text{Nbhd}(d)|}}. \tag{8}$$

And the item-based collaborative filtering score of item b is

$$\text{Score}_{\text{ICF}}(b|x_\ell) = \sum_{d \in x_\ell} \text{sim}_{\text{item}}(b, d). \tag{9}$$

In addition to these two baselines, we consider the extremely simple baseline of ranking items by their frequency in the training set. We call this the frequency baseline.

3 Name Recommendations with the Nameling Dataset

We now demonstrate our similarity-weighted adjusted confidence measure on the Nameling dataset released for the ECML PKDD 2013 Discovery Challenge. We also compare the alternative confidence measures and baseline methods from Section 2. A description of the Nameling dataset can be found in [12], and details about the challenge task can be had in the introduction to these workshop proceedings. For the sake of self-containment, we give a brief description here.

3.1 The Nameling Public Dataset

The dataset contains the interactions of users with the Nameling website `http://nameling.net`, a site that allows its users to explore information about names and provides a list of similar names. A user enters a name, and the Nameling system provides a list of similar names. Some of the similar names are given category descriptions, like "English given names," or "Hypocorisms." There are five types of interactions in the dataset: "ENTER_SEARCH," when the user enters a name into the search field; "LINK_SEARCH," when the user clicks on one of the listed similar names to search for it; "LINK_CATEGORY_SEARCH," when the user clicks on a category name to list other names of the same category; "NAME_DETAILS" when the user clicks for more details about a name; and "ADD_FAVORITE" when the user adds a name to his or her list of favorites. The dataset contains 515,848 interactions from 60,922 users.

The data were split into training and test sets by, for users with sufficiently many "ENTER_SEARCH" interactions, setting the last two "ENTER_SEARCH" interactions aside as a test set. Some other considerations were made for duplicate entries - see the introduction to the workshop proceedings for details. The end result was a training set of 443,178 interactions from the 60,922 users, and a test set consisting of the last two "ENTER_SEARCH" names for 13,008 of the users. The task was to use the interactions in the training set to predict the two

names in the test set for each of the test users by producing for each test user a ranked list of recommended names. The evaluation metric was mean average precision of the first 1000 recommendations - see the proceedings introduction for more details.

3.2 Data Pre-processing

We did minimal data pre-processing, to highlight the ability of similarity-weighted adjusted confidence to perform well without carefully crafted features or manual consideration of special cases. We discarded users with no "ENTER_SEARCH" interactions, which left 54,439 users. For each user i, we formed the set of items x_i as "name, interaction type" for all interactions from that user. For example, "Primrose, ENTER_SEARCH" was the feature indicating that the user did an "ENTER_SEARCH" for the name Primrose. The total feature collection Z contained "name, interaction type" for all of the entries in the interaction database. The total number of items in Z was $n = 34,070$. No other data pre-processing was done.

To form rules, we took as left-hand sides a all individual interaction entries: $\mathcal{A} = Z$. We considered as right-hand sides b all valid names to be recommended (among other things, this excluded names that were previously entered by that user - see the proceedings introduction for details on which names were excluded from the test set). An example rule is "Primrose, ENTER_SEARCH $\rightarrow$ Katniss."

3.3 Results

We applied confidence, adjusted confidence, similarity-weighted confidence, and similarity-weighted adjusted confidence to the training set to generate recommendations for the test users. For the adjusted measures, we found the best performance on the test set with $K = 4$ for similarity-weighted adjusted confidence and $K = 10$ for adjusted confidence, as shown in Figure 1. We also applied the user-based collaborative filtering, item-based collaborative filtering, and frequency baselines to generate recommendations. For all of these recommender system approaches, the mean average precision at 1000 on the test set is shown in Table 1.

Similarity-weighted adjusted confidence gave the best performance, and similarity weighting led to a 4.2% increase in performance over (unweighted) adjusted confidence. The adjustment also led to a 9.7% increase in performance from similarity-weighted confidence to similarity-weighted adjusted confidence. User-based collaborative filtering performed well compared to the frequency baseline, but was outperformed by similarity-weighted adjusted confidence by 11.4%. Item-based collaborative filtering performed very poorly.

An advantage of using association rules as opposed to techniques based in regression or matrix factorization is that there is no explicit error minimization problem being solved. This means that association rules generally do not have the same propensity to overfit as algorithms based in empirical risk minimization.

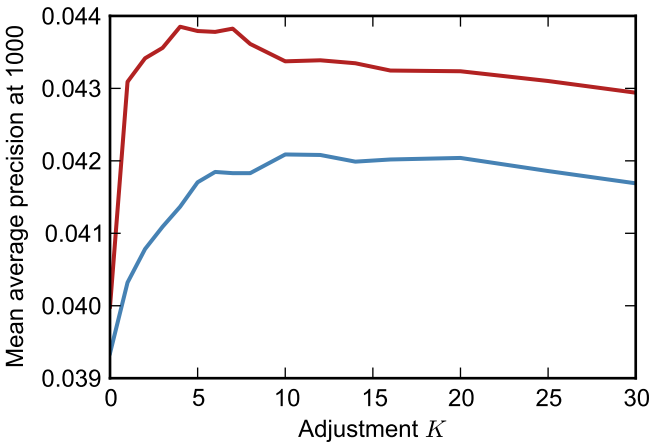


Fig. 1. Test performance for adjusted confidence (blue) and similarity-weighted adjusted confidence (red) for varying amounts of adjustment K.

Table 1. Mean average precision at 1000 for the recommender system approaches discussed in the paper.

Recommender system	Mean average precision
Similarity-weighted adjusted confidence, $K = 4$	0.04385
Adjusted confidence, $K = 10$	0.04208
Similarity-weighted confidence	0.03998
User-based collaborative filtering	0.03936
Confidence	0.03934
Frequency	0.02821
Item-based collaborative filtering	0.01898

We found that the performance on the discovery challenge hold-out dataset was similar to that which we measured on the public test set in Table 1.

4 Conclusions

Similarity-weighted adjusted confidence is a natural fit for the Nameling dataset and the name recommendation task. First, the dataset is extremely sparse (see [12]). The Bayesian adjustment K increases performance by reducing variance for low-support itemsets, and this dataset contains many low-support yet informative itemsets. Second, preferences for names are very heterogeneous. Incorporating the similarity weighting from user-based collaborative filtering into the confidence measure helps to focus the estimation on the more informative users.

Association rules and similarity-weighted adjusted confidence are powerful tools for creating a scalable and interpretable recommender system that will perform well in many domains.

Acknowledgments. Thanks to Stephan Doerfel, Andreas Hotho, Robert Jäschke, Folke Mitzlaff, and Juergen Mueller for organizing the ECML PKDD 2013 Discovery Challenge, and for making their excellent Nameling dataset publicly available. Thanks also to Cynthia Rudin for support and for many discussions on using rules for predictive modeling.

References

1. Zaki, M.J., Ogihara, M.: Theoretical foundations of association rules. In: 3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (1998)
2. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM conference on Computer Supported Cooperative Work. pp. 241–250. CSCW '00 (2000)
3. McSherry, D.: Explanation in recommender systems. Artificial Intelligence Review 24(2), 179–197 (2005)
4. Herlocker, J.L., Konstan, J.A., Terveen, L.G., John, Riedl, T.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems 22, 5–53 (2004)
5. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. pp. 43–52. UAI'98 (1998)
6. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web. pp. 285–295. WWW '01 (2001)
7. Rudin, C., Letham, B., Salleb-Aouissi, A., Kogan, E., Madigan, D.: Sequential event prediction with association rules. In: Proceedings of the 24th Annual Conference on Learning Theory. pp. 615–634. COLT '11 (2011)
8. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 226–239 (1998)
9. Tax, D.M., Breukelen, M.V., Duin, R.P., Kittler, J.: Combining multiple classifiers by averaging or by multiplying? Pattern Recognition 33, 1475–1485 (2000)
10. Letham, B., Rudin, C., Madigan, D.: Sequential event prediction. Machine Learning (2013), in press
11. Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 271–278. SIGIR '07 (2007)
12. Mitzlaff, F., Stumme, G.: Recommending given names (2013), `http://arxiv.org/abs/1302.4412`, preprint, arxiv:1302.4412

Factor Models for Recommending Given Names

Immanuel Bayer and Steffen Rendle

University of Konstanz, 78457 Konstanz, Germany,
{immanuel.bayer, steffen.rendle}@uni-konstanz.de

Abstract. We describe in this paper our contribution to the ECML PKDD Discovery Challenge 2013 (Offline Track). This years task was to predict the next given names a user of a name search engine interacts with. We model the user preferences with a sequential factor model that we optimize with respect to the Bayesian Personalized Ranking (BPR) Optimization Criterion. Therefore we complement the sequential factor model with prefix smoothing in order to explicitly model syntactical similarity.

Keywords: factor model, given names, recommender system

1 Introduction

We describe in this paper our contribution to the ECML PKDD Discovery Challenge 2013 (Offline Track). This years task was to recommend given names to user of the name search engine "Nameling"[1] based on their historical search and clicking behavior. We interpret the problem as a classical recommender problem and use a purely statistical approach based on the user history. No meta data such as word similarity lists or geographic information for the users are used.
We use a factorized personalized Markov Chain (FPMC) [4] model in order to capture the user specific name preferences. The Bayesian Personalized Ranking (BPR) Optimization Criterion [3] is used to learn the latent variables of this model. We complement this factor model with syntactical similarity information by applying prefix smoothing to the name ranking.

2 Terminology and Formalization

Let $U = \{u_1, u_2, \ldots\}$ be the set of all users and $N = \{n_1, n_2, \ldots\}$ the set of all names. We further use $T = \mathbb{N}$ to identify user action over time.

The scoring function

$$\hat{y} : U \times T \times N \to \mathbb{R} \tag{1}$$

returns the estimated preference of a user U at time T for a name N. To obtain the user specific ranking for every user $u \in U$, each name is scored – i.e. $\hat{y}(u, t, n_1)$, $\hat{y}(u, t, n_2)$, etc. is computed – and the names are sorted by their score. We further define $D \subseteq U \times T \times N$ as the training set of available names which have been observed in the logs.

3 Sequential Factor Model

In order to extract training samples from the user activities, we first defined four indicator functions. These are then used to encode the training samples as sparse real valued features that can be used in a factorization machine. Finally, we give a formal definition of our prefix smoothing approach.

3.1 Indicators

Our model assumes that the name preference of a user u at time t can be explained by:

1. The ID of the user: u.
2. The ID of the name: n.
3. The **l**ast name selected by the user: $l : U \times T \to N$.
4. The **h**istory of all names selected by the user up to time t: $h : U \times T \to \mathcal{P}(N)$.

Besides these four indicators, the model should also take into account all interactions between indicators. E.g. the interaction between name n and last name $l(u, t)$ would model the effect for choosing name n if the name $l(u, t)$ has been selected before. In total, the first three indicators correspond to a personalized Markov chain [4]. The forth indicator can be seen as a Markov chain with long memory where all the history is aggregated into a single set.

3.2 Factorization Machine

The number of pairwise interactions between variables is high and cannot be estimated reliably with standard parametrization. Thus, we use a factorized parametrization [4] which allows to estimate parameters even in highly sparse data.

The ideas described so far can be realized with a factorization machine [2]. For this purpose, the four indicator variables are translated into a sparse real valued feature vector $\mathbf{x} \in \mathbb{R}^p$ with $p = |U| + |N| + |N| + |N|$ many predictor variables. The standard encoding described e.g. in [2] is used.

For example, for a case (u, t, n) let the values of the four indicators be:

1. user ID: 0,
2. name to rank: *Anna*,
3. last name selected by user: *Jana*,
4. history of all names selected by the user up to time t: {*Petra, Annabelle, Maria*}.

This can be encoded as a real valued feature vector of the form

$$\mathbf{x}(u, t, n) = (\underbrace{1, \ldots, 0}_{|U|}, \underbrace{0, 1, 0, \ldots}_{|N|}, \underbrace{0, \ldots, 1, 0, \ldots}_{|N|}, \underbrace{0, 0.33, \ldots, 0.33, \ldots, 0.33, \ldots 0}_{|N|}). \tag{2}$$

The factorization machine (FM) model [2] of order $d = 2$ can be applied to the generated feature vector $\mathbf{x}$ and reads

$$\hat{y}^{\text{FM}}(\mathbf{x}(u,t,n)) := w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} x_j x_{j'} \sum_{f=1}^{k} v_{j,f} v_{j',f} \tag{3}$$

Here, $w_0 \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^p, V \in \mathbb{R}^{p \times k}$ are the model parameters, $k \in \mathbb{N}$ is the size/ dimensionality of the latent space. Thus, the model has one feature vector $\mathbf{v}_i$ for each variable x_i.

Empirically inspecting the generated recommendations with this model shows (see Table 2) that the semantic meaning of names is found: e.g. if a user searches mainly for female names, female names are recommended; if the user selects typically short names, short ones are recommended, etc. The reason for the success is that the model automatically finds latent features $\mathbf{v}_n \in \mathbb{R}^k$ for each name n which describe its characteristics. Such characteristics could be gender, name length, etc.

3.3 Prefix Smoothing

In general, the proposed model can express any kind of pairwise relation between names[1]. However, under small data sizes, the model might have problems to find relations between infrequent names. To make an example, the model can express and will automatically learn that *Anna* and *Anne* are syntactically similar or that *Farid* and *Behrouz* are semantically similar (both Persian masculine names) if the data logs are large enough. However, the size of the observed logs is limited and the model cannot learn all these relations reliably – especially not for infrequent names.

To overcome the problem, we inject some syntactical relations manually into the model. We create indicators stating that two names (name n and last name $l(u,t)$) share a prefix of length m – we consider prefix lengths of $m \in \{1,2,3,4,5,6\}$. We add these indicators about syntactical similarity to the FM model mentioned above:

$$\hat{y}(u,t,n) := \hat{y}^{\text{FM}}(\mathbf{x}(u,t,n)) + \sum_{m \in \{1,2,3,4,5,6\}} z_m \, \delta(\text{prefix}(n,m) = \text{prefix}(l(u,t),m)), \tag{4}$$

where δ is the indicator function – i.e. $\delta(b) = 1$ if b is true – and $\text{prefix}(s,m)$ returns the prefix of string s of length m.

The final model is slightly more complex and considers also syntactical similarity with the next-to-last name:

$$\hat{y}(u,t,n) := \hat{y}^{\text{FM}}(\mathbf{x}(u,t,n)) + \sum_{t' \in \{t,t-1\}} \sum_{m \in \{1,2,3,4,5,6\}} z_{m,t'} \, \delta(\text{prefix}(n,m) = \text{prefix}(l(u,t'),m)). \tag{5}$$

[1] Note that semantic or syntactical similarities are also just pairwise relations.

Please note that the idea of prefix smoothing – i.e. the indicator $\delta(\text{prefix}(n, m) = \text{prefix}(l(u, t), m))$ – can be used directly in the FM by enlarging the feature vector $\mathbf{x}$. Using this representation, the parameters $z_{m,t}$ are part of the $\mathbf{w}$ parameters of the original FM. Extending the prefix smoothing to longer prefixes as well as taking into account names earlier then $t - 1$ could further improve the model.

4 Learning

We have a lot of positive samples if we assume that the user likes names he interacts with but we know little about other names. Encoding all names the user didn't interact with as negative samples can introduce wrong user preferences since we can not distinguish between a name the user knowingly ignored and a name the user has never seen. We avoid this problem by using a pairwise loss function.

4.1 Optimization Criterion

The model parameters of the FM are learned by discriminating between previously selected and unselected names. This optimization criterion has been proposed for item recommendation as BPR (*Bayesian Personalized Ranking*) [3] and has been used in several other recommendation tasks including sequential recommendation [4]. For the task of name recommendation it reads:

$$\text{BPR-OPT} := \sum_{(u,t,n) \in D} \sum_{n_2 \in (N \setminus \{n\})} \ln \sigma(\hat{y}(u, t, n) - \hat{y}(u, t, n_2)) - \lambda_\Theta ||\Theta||^2 \quad (6)$$

where $\hat{y}$ is the FM (using the predictor variables encoded in the real valued vector $\mathbf{x}$) and Θ is a vector containing the model parameters $V, \mathbf{w}, w_0$.

4.2 Algorithm

The standard BPR algorithm [3] is a stochastic gradient descent (SGD) algorithm. The algorithm samples first a positive observation $(u, t, n) \in D$ and then a negative name n_2 uniformly from $N \setminus \{n\}$. A gradient step is done on this pairwise comparison. In our implementation, instead of using a uniform distribution for negative names, the names are sampled approximately proportional to their expected rank.

Even though FM parameters and prefix smoothing could be learned jointly with BPR, for simplicity[2] we learned only the FM parameters with BPR and selected the prefix smoothing parameters (only 12 parameters) manually.

[2] The reason for separating both parts were only of practical matter because we reused an existing implementation.

4.3 Ensembling Factor Models

The BPR algorithm is a point estimator and returns one ranking. We consider uncertainty in the ranking by running the learning algorithm three times, each time with a different sampling hyperparameter. While training each model, we select[3] in total 20 iterations for which we predict the ranking each – i.e. we have 20 (slightly) different scores $\hat{y}(u, t, n)$ for each triple (u, t, n). The final scoring function is an unweighted average of the 20 scoring functions.

5 Experimental Results

In this section, we first present our scores from the official leaderboard and then discuss the latent features learned by our model on randomly selected samples.

5.1 Evaluation on Holdout Set

We separate the training data into a new validation and training set using the splitting script provided. This gives us a training set with 60.922 user and a test set with 13.008 user. We use the first 3.000 user from the new test set to calibrate our models.

5.2 Results

Table 1 shows the results we achieved on the official leaderboard. The table contains also the scores that we obtained on our validation set. The score difference between validation and challenge data might be partially explained by the different ensemble size that we used in the two evaluations. We ensemble 20 rankings for the last submitted model (as described in section 4.3) but an ensemble of 100 rankings for our validation set evaluation. After some last minute changes we had only time to select 20 rankings for the final submission.

Table 1. Results as reported on the challenge leaderboard and the official evaluation script (modified MAP@1000).

Model	validation	challenge
most-popular-item	0.0294	0.0259
FM-ens (with prefix smoothing)	0.0550	0.0472

[3] The selected iterations were chosen close to the best iterations on the holdout set, i.e. slightly before and after the best iteration.

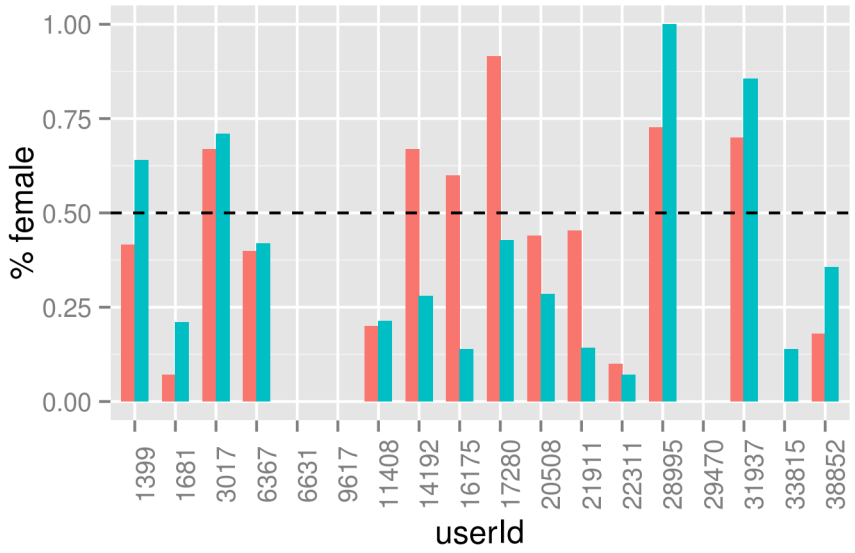


Fig. 1. Each bar pair represents the percentage of female names in the user history $h(u,t)$ (left) and the top 14 recommended names (right). The users evaluated here are the same as listed in Table 2. Similar height of both bars (left and right) for one user indicates that the user-preferred ratio of female to male names is closely reflected in the recommendations.

5.3 Ranking Analysis

In this section, we demonstrate on samples that our model is able to represent name similarities and user preferences through its latent variables. We illustrate this on name characteristics such as gender or length because the effects of those items are easy to recognize. More subtle effects might also be captured by our model but an casual inspection on a small sample might not suffice to identify them.

We select a subset of users from the challenge test set that have at least 30 and at most 50 activities (This is done to avoid users without history and to save space by avoiding user with a very large history). From this set, we randomly select 18 users and list them in table 2. The first line $h(u,t)$ for each user lists all names in his user history. Duplicates and names that are not used in the competition (not listed in namelist.txt) have been removed. The second line lists the top 14 recommendations generated by our FM model[4].

[4] The number of listed recommendations and the number of randomly selected users are adjusted to the available space.

Table 2. Top 14 recommended names, for a random selection of users. Top14 shows the list of recommended names, where the leftmost name is the top recommended one. $h(u,t)$ are all names in the user history up to the last available time t. The last name selected by the user is pried in bold.

userId	set	names
3017	$h(u,t)$	tanja, birgit, antje, karin, heide, kevin, maria, arthur, , mohammed, celine, cem, **mia**
	top14	emma, anna, julia, johanna, michael, katharina, marie, thomas, sophie, eva, alexander, christian, charlotte, lena
1681	$h(u,t)$	roman, boris, leon, robin, david, stikar, krasimira, julian, fabian, erik, henrik, **bastian**
	top14	emma, daniel, alexander, thomas, anna, david, christian, andreas, michael, sebastian, paul, jan, julia, kevin
9617	$h(u,t)$	tobias, lasse, niklas, malte, anton, **justus**
	top14	malte, jan, arne, ole, mats, nils, tim, lars, mika, finn, ben, paul, matti, linus
22311	$h(u,t)$	jakob, karin, paula, charlotte, luise, miriam, rosa, caroline, olivia, **marie**
	top14	emma, pauline, greta, anna, luisa, julia, lena, marie, paul, johanna, mathilda, ida, maria, frieda
38852	$h(u,t)$	alma, reto, christoph, philipp, paul, peter, remo, karl, sarah, lukas, **alex**
	top14	emma, anna, alexander, julia, katharina, johanna, sebastian, lena, michael, marie, hannah, maximilian, jakob, elisabeth
17280	$h(u,t)$	ida, lotta, lena, sara, charlotte, emma, greta, nora, leo, leonie, emilie, **mara**
	top14	emil, greta, paul, emma, theo, frieda, anna, oskar, anton, paula, jakob, ben, moritz, julia
1399	$h(u,t)$	verena, kay, anni, eva, effi, friedrich, friedolin, xavier, james, paulchen, resi, **paul**
	top14	emma, anna, michael, thomas, christian, stefan, mia, alexander, andreas, markus, marie, daniel, greta, paul
21911	$h(u,t)$	mia, claudia, jasmin, nicolas, niklas, andris, leinis, lara, lavina, ben, **kilian**
	top14	emma, sophie, laura, julia, anna, mia, leon, lena, david, lea, sarah, luca, sophia, lisa
29470	$h(u,t)$	lasse, kjell, bodo, enno, benno, luis, mads, mats, **caspar**
	top14	louis, henri, emil, ben, levi, finn, paul, leo, ole, max, jonas, anton, till, noah
11408	$h(u,t)$	clemens, paul, anton, matthias, **isabel**
	top14	emma, anna, emil, jakob, alexander, andreas, thomas, julia, michael, moritz, felix, peter, jonas, johannes
28995	$h(u,t)$	noa, luitgard, heidrun, yeni, nova, zoe, naemi, sune, ruben, cano, **jon**
	top14	mila, emma, luca, mia, laura, lilith, mara, sarah, leonie, lea, julia, anna, sophie, sandra
6367	$h(u,t)$	jonas, anna, esther, anja, maris, linda, lorenz, lennard, jannes, **gerhard**
	top14	emma, anna, paul, julia, emil, greta, alexander, jakob, david, daniel, jonas, katharina, johanna, anton
33815	$h(u,t)$	raphael, yannick, timon, jol, rafael, neo, nearchos, iason, jason, **phineus**
	top14	daniel, alexander, thomas, david, sebastian, emma, christian, michael, anna, andreas, leon, jonas, kevin, elias
16175	$h(u,t)$	mirka, liane, irina, vincent, **carlos**
	top14	alexander, thomas, daniel, michael, andreas, christian, david, peter, sebastian, anna, emma, martin, markus, maximilian
6631	$h(u,t)$	tim, ellis, lino, nino, paul, leon, leonard, **tobi**
	top14	luis, ben, luca, finn, noah, jonas, max, julian, leo, lukas, tom, maximilian, jan, mika
14192	$h(u,t)$	jasmin, maja, matthias, felix, emilia, **luisa**
	top14	emma, paul, leon, anna, daniel, thomas, sebastian, moritz, alexander, simon, christian, julia, mia, max
31937	$h(u,t)$	ida, kalle, melanie, mara, katja, merle, nele, juno, linda, **marlen**
	top14	greta, emma, lotta, lina, mia, anna, ella, emil, frieda, lotte, charlotte, lasse, frida, julia
0508	$h(u,t)$	luisa, anna, bettina, emilia, yngwie, oskar, emil, albert, **frederick**
	top14	anton, paul, jakob, emma, moritz, theo, johann, karl, julia, julius, greta, paula, felix, jonathan

5.4 Discussion of Learned Effects

Even though the predictive strength is best jugged by the leaderboard score, we want to give an impression of the kind of relationships that our model has learned from the training data. Please note that the rankings presented here are without prefix smoothing. This means that the model had no information on how long or which characters a certain name contains since names are only represented by unique identifiers.
Our model learns to distinguish between **female** and **male** names and recognizes if a user prefers male or female names. For users that have very strong preferences, such as user 6631, 9617, 29470, 3017, 28995, the recommendations that are accordingly balanced (see Table 2 and Figure 1). For users with very few user activities this becomes less reliable as can be seen on users; 16175 and 14192.
The model also learned if a name is **long** (user 3017, 38852, 16175) or very **short** (user 9617, 29470). It also recognizes if a user prefers **infrequent** names User 28995 for example has *kjell* and *enno* in his history and gets names like *levi* and *finn* recommended. **Double consonant** names are surprisingly frequent in the recommendations for user 31937, while names with **similar prefixes** have similar ranks for user 16175 (*martin, markus, maximilian*) or user 20508 (*julia, julius*) and also *lotte* and *charlotte* are ranked next to each other for user 31937. A more detailed analysis of the learned ranking could reveal more information on how people judge the similarities of given names.

6 Conclusion

In this paper, we have shown that a Factorization Machine [2] is well suited for the task of recommending given names. When the model parameters are optimized with respect to the personalized ranking criterion BPR-Opt [3], the latent variables are able to express name preferences such as name length and gender. Is important to note that this information is not part of the model input. Being able to learn this characteristic is useful if this information is not readily available.

The data used in this competition were strongly dominated by German speaking users (according to the user location obtained from the ip addresses) but our approach is supposed to work equally well with data that contain names from different alphabets or a user base that contains strong regional preferences without any modification.

We like to point out the close relation of given name recommendation to tasks such as movie recommendation where factor models have been very successful. The latent variables in this competition represent here syntactic and semantic similarity instead of movie related characteristics like genres or common actor. We have further shown that for infrequent names and small data sets the injection of regularizing information such as syntactical similarity does improve the prediction quality. We however expect that the benefit of information injection diminishes with the size of available data.

7 ACKNOWLEDGMENTS

We gratefully acknowledge funding by *Baden-Württemberg Stiftung*.

References

1. Mitzlaff, F., Stumme, G.: Namelings - discover given name relatedness based on data from the social web. In Aberer, K., Flache, A., Jager, W., Liu, L., Tang, J., Guret, C., eds.: SocInfo. Volume 7710 of Lecture Notes in Computer Science., Springer (2012) 531–534
2. Rendle, S.: Factorization machines with libFM. ACM Trans. Intell. Syst. Technol. **3**(3) (May 2012) 57:1–57:22
3. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI 2009). (2009)
4. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized markov chains for next-basket recommendation. In: WWW '10: Proceedings of the 19th international conference on World wide web, New York, NY, USA, ACM (2010) 811–820