# Modeling User Interests for Zero-query Ranking

Liu Yang[1]*, Qi Guo[2], Yang Song[3], Sha Meng[2], Milad Shokouhi[4], Kieran McDonald[2], and W. Bruce Croft[1]

[1] Center for Intelligent Information Retrieval, University of Massachusetts Amherst, MA, USA
[2] Microsoft Bing, Bellevue, WA, USA
[3] Microsoft Research Redmond, WA, USA
[4] Microsoft Research Cambridge, Cambridge, UK
{lyang, croft}@cs.umass.edu,{qiguo, yangsong, shmeng, milads, kieran.mcdonald}@microsoft.com

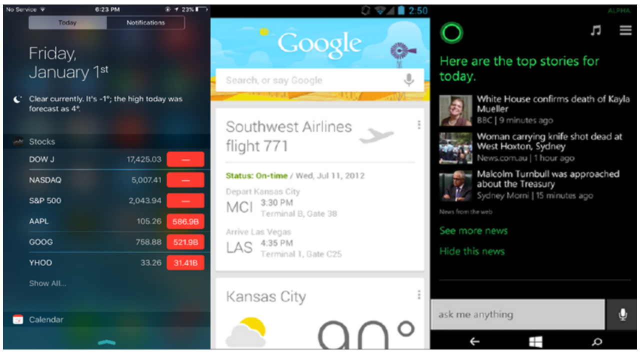
**Abstract.** Proactive search systems like Google Now and Microsoft Cortana have gained increasing popularity with the growth of mobile Internet. Unlike traditional reactive search systems where search engines return results in response to queries issued by the users, proactive systems actively push information cards to the users on mobile devices based on the context around time, location, environment (e.g., weather), and user interests. A proactive system is a zero-query information retrieval system, which makes user modeling critical for understanding user information needs. In this paper, we study user modeling in proactive search systems and propose a learning to rank method for proactive ranking. We explore a variety of ways of modeling user interests, ranging from direct modeling of historical interaction with content types to finer-grained entity-level modeling, and user demographical information. To reduce the feature sparsity problem in entity modeling, we propose semantic similarity features using word embedding and an entity taxonomy in knowledge base. Experiments performed with data from a large commercial proactive search system show that our method significantly outperforms a strong baseline method deployed in the production system.

## 1 Introduction

The recent boom of mobile internet has seen the emergence of proactive search systems like Google Now, Apple Siri and Microsoft Cortana. Unlike traditional reactive Web search systems where the search engines return results in response to queries issued by users, proactive search systems actively push information cards to users on mobile devices based on the context such as time, location, environment (e.g., weather), and user interests. Information cards are concise and informative snippets commonly shown in many intelligent personal assistant systems. Figure 1 shows examples of proactive information cards presented in Apple Siri (stocks), Google Now (flight and weather) and Microsoft Cortana (news). There are no explicit queries for these returned information cards which are triggered by some particular context.

The need for these proactive search systems increases in mobile environments, where the users' ability to interact with the system is hampered by the physical limita-

* Work primarily done when interning at Microsoft.



Friday, January 1st

Clear currently. It's -1°; the high today was forecast as 4°.

| | | |
|---|---|---|
| DOW J | 17,425.03 | — |
| NASDAQ | 5,007.41 | — |
| S&P 500 | 2,043.94 | — |
| AAPL | 105.26 | 586.9B |
| GOOG | 758.88 | 521.9B |
| YHOO | 33.26 | 31.41B |

Southwest Airlines flight 771

Status: On-time / Wed, Jul 11, 2012

Depart Kansas City
MCI 3:30 PM
Terminal B, Gate 38

Arrive Las Vegas
LAS 4:35 PM
Terminal 1, Gate C25

Kansas City
90°

Here are the top stories for today.

White House confirms death of Kayla Mueller
BBC | 9 minutes ago

Woman carrying knife shot dead at West Hoxton, Sydney
News.com.au | 1 hour ago

Malcolm Turnbull was approached about the Treasury
Sydney Morni | 15 minutes ago

Fig. 1: Examples of proactive information cards presented in Apple Siri (stocks), Google Now (flight and weather) and Microsoft Cortana (news).

tions of the devices[2]. Thus future information retrieval systems must infer user information needs and respond with information appropriate to the current context without the user having to enter a query. User modeling plays a critical role for understanding user information needs in the absence of query information.

Despite the abundance of research about user modeling and personalization for reactive Web search systems [1,4,19], little is known about how to model user interests to improve the effectiveness of the proactive systems. The closest research to our paper is the recent work by Shokouhi and Guo [15] , where the authors looked at the usage patterns of proactive systems and aimed to understand the connections between reactive searching behavior and user interaction with the proactive systems.

In this paper, we explore a broader variety of ways and data sources in modeling user interests and compare the proposed method with a baseline that is similar and comparable to the Carre model presented in [15] for the application of improving ranking of information cards of the proactive systems. Our explorations include modeling at the coarser-level of card types, to finer-grained modeling of entities, and the capturing of variations lie in the variety of demographics. We develop entity based representations for user interests and card topics. Entities are extracted from user search/browse logs across multiple commercial platforms and devices to represent user interests. For card topics, entities are extracted from the associated URLs. To reduce the feature sparsity problem, we propose entity semantic similarity features based on word embedding and an entity taxonomy in knowledge base. The contributions of our work can be summarized as the follows:

- We present the first in-depth study on modeling user interests for improving the ranking of proactive search systems.
- We propose a variety of approaches of modeling user interests, ranging from coarser-grained modeling of card type preferences, to finer-grained modeling of entity preferences, and the variations in demographics.
- We perform a thorough experimental evaluation of the proposed methods with large-scale logs from a commercial proactive system, showing that our method significantly outperforms a strong baseline ranker deployed in the production system.

- We conduct in-depth feature analysis, which provides insights for guiding future feature design of the proactive ranking systems.

## 2 Related Work

There is a range of previous research related to our work that falls into different categories including proactive information retrieval, information cards, search personalization and recommender systems.

**Proactive Information Retrieval.** Rhodes and Maes [14] proposed the just-in-time information retrieval agent (JITIR agent) that proactively retrieves and presents information based on a person's local context. The motivation of modern proactive search system is very similar to JITIRs, but the monitored user context and presented content of modern proactive system are more extensive than traditional JITIRs.

**Information Cards.** Web search has seen rapid growth in mobile search traffic, where answer-like results on information cards are better choices than a ranked list to address simple information needs given the relative small size of screens on mobile devices. For some types of information cards like weather, users could directly find the target information from contents on cards without any clicks. This problem was introduced by Li et al. [9] as "good abandonment". Based on this problem, Guo et al. [7] proposed a study of modeling interactions on touch-enabled device for improving Web search ranking. Lagun et al. [8] studied the browser viewport which is defined as visible portion of a web page on mobile phones to provide good measurement of search satisfaction in the absence of clicks. For our experiments, we also consider viewport based dwell time to generate relevance labels for information cards to handle the good abandonment problem.

**Search Personalization.** Proactive systems recommend highly personalized contents to users based on their interest and context. Hence, our work is also related to previous research on search personalization [21,19,6]. Fox et al. [6] showed there was an association between implicit measures of user activity and the user's explicit satisfaction ratings. Agichtein et al. [1] showed incorporating implicit feedback obtained in a real web search setting can improve web search ranking. Bennett et al. [4] studied how short-term (session) behavior and long-term (historic) behavior interact, and how each may be used in isolation or in combination to optimally contribute to gains in relevance through search personalization. We also consider implicit feedback features based on user interactions with different card types and compare the relative importance of this feature group with other feature groups like entity based user interests features for proactive ranking.

**Recommender Systems.** Similar to recommender systems [13], we also push the most relevant content to the user based on user personal interests without a query issued by the user. However, the recommended items in the proactive system are for a smaller set of items that are highly heterogeneous and need to be personalized and contextualized in the ranking [15]. Unlike collaborative filtering methods [17] commonly used in recommender systems, we adopt a learning to rank framework that is suitable for combining multiple features derived from various user history information.

## 3 Method Overview

We adopt common IR terminology when we define the proactive search problem. A proactive impression consists of a ranked list of information cards presented to users together with the user interaction logs recording clicks and viewports. Given a set of information cards $\{C_1, C_2, ..., C_n\}$ and the corresponding relevance labels, our task is to learn a ranking model ***R*** to rank the cards based on available features $\theta$ and optimize a pre-defined metric $E$ defined over the card ranked list.

We propose a framework for proactive ranking referred to as **UMPRanker** (User Modeling based Proactive Ranker). Firstly we mine user interests from multiple user logs. Each user is distinguished by a unique and anonymized identifier which is commonly used in these platforms. The information collected from these different platforms forms the basis of our user modeling. Then we derive multiple user interest features including entity based user interests, card type based implicit feedback and user demographics based on the collected information. Information cards are generated from multiple pre-defined information sources and templates including weather, finance, news, calendar, places, event, sports, flight, traffic, fitness, etc. We also extract card features from the associated URLs and card types. Given user features and card features, we can train a learning to rank model. Given a trigger context like particular time, location or event, information cards are ranked by the model and pushed to the user's device.

## 4 Mining User Interests From Logs

We can derive user interests from the short text that users specified and the textual content that users engaged with during their historical activity. Specifically, the information sources of user interests we consider include the following user behavior in logs:

1. Issued queries from the search behavior.
2. Satisfactory(SAT) clicked documents from the search behavior.
3. Browsed documents on an Internet browser and a Web portal.
4. Clicks and viewports on a personal assistant.

Note that users have the right to choose whether they would like the services to collect their behavior data. The logs we collected are from "opt-in" users only. To represent user interests, we extract entities from the text content specified by user behaviors. We can also represent information card topics by entities exacted from card URLs. Entities in user profiles and cards are linked with entities in a large scale knowledge base to get a richer representation of user interests.

## 5 Ranking Feature Extraction

### 5.1 Card Type based Implicit Feedback Features (IF)

This feature group is based on statistics of user interactive history with different card types like average view time of each card type, accept ratio of each card type, etc. This group of features aims at capturing individual user preferences of particular card type, for example, news, based on the statistics of the historical interactions. Specifically, for each <user, card type> pair, features extracted include historical clickthrough rate (CTR), SAT CTR (i.e., clicks with more than 30 seconds dwell time on landing pages), SAT view rate (i.e., card views with more than 30 seconds), hybrid SAT rate (i.e., rate of either a SAT click or a SAT view), view rate, average view time, average view speed, accept ratio of the card suggestions, untrack ratio of the card type, ratio of the cards being a suggestion. The details of these features are explained in Table 1.

Table 1: Summary of card type based implicit feedback features (IF).

| Feature | Description |
|---|---|
| CTR | Personal historical clickthrough rate of the card type. |
| SATCTR | Personal historical SAT (landing page dwell time $>$ 30 seconds) clickthrough rate of the card type. |
| SATViewRate | Personal historical SAT (card view time $>$ 30 seconds) view rate of the card type. |
| ViewRate | Personal historical view rate of the card type. |
| AverageViewTime | Personal historical average view time of the card type. |
| AverageViewTimeSpeed | Personal historical average view time per pixel of the card type. |
| AcceptRatio | Personal historical accept ratio when the card type was presented as a suggestion. |
| UnTrackRatio | Personal historical ratio untrack the card type. |
| SuggestionRate | Personal historical ratio of seeing the card type being presented as a suggestion. |

### 5.2 Entity based User Interests Features (EF)

As described in Section 4, we can represent user interests by entities extracted from user behavior across multiple services and devices. For cards with URLs, we can also represent card topics with entities. So the next problem is how to measure the similarity between user entity sets and card entity sets. We consider features including exact match, term match, language models, word embedding and entity taxonomy in the knowledge base. In the following parts, we let $U_i$ and $C_j$ denote the entity set of the $i$-th user and the $j$-th card.

Table 2: Summary of entity based user interests features (EF).

| Feature | Description |
|---|---|
| RawMatchCount | The raw match count of entities by id in $U_i$ and $C_j$. |
| EMJaccardIndex | The Jaccard Index of entities matched by id in $U_i$ and $C_j$. |
| TMNoWeight | The cosine similarity between two entity term distributions in $U_i$ and $C_j$. |
| TMWeighted | Similar to TMNoWeight, but terms are weighted by impression count. |
| LMScore | The log likelihood of generating terms in $C_j$ using a language model constructed from terms in $U_i$. |
| WordEBDMin | The similarity between $U_i$ and $C_j$ based on word embedding features(single-link). |
| WordEBDMax | The similarity between $U_i$ and $C_j$ based on word embedding features(complete-link). |
| WordEBDAvgNoWeight | The similarity between $U_i$ and $C_j$ based on word embedding features(average-link-noWeight). |
| WordEBDAvgWeighted | The similarity between $U_i$ and $C_j$ based on word embedding features(average-link-weighted). |
| KBTaxonomyLevel1 | The similarity between $U_i$ and $C_j$ based on entity taxonomy similarity in level 1. |
| KBTaxonomyLevel1Weighted | Similar to EntityKBTaxonomyLevel1 but each entity is weighted by its impression counts. |
| KBTaxonomyLevel2 | The similarity between $U_i$ and $C_j$ based on entity taxonomy similarity in level 2. |
| KBTaxonomyLevel2Weighted | Similar to EntityKBTaxonomyLevel2 but each entity is weighted by its impression counts. |

**Exact Match**. The first feature is *exact match*. It is computed based on the number of common entities matched by entity ID in $U_i$ and $C_j$ . We consider two variations:

*RawMatchCount* and *EMJaccardIndex*. *RawMatchCount* uses the original match count as the feature value. *EMJaccardIndex* computes the Jaccard Index of $U_i$ and $C_j$.

**Term Match**. *exact match* feature suffers from feature sparsity problem. A better method is to treat $U_i$ and $C_j$ as two term sets. Then we could get two entity term distributions over $U_i$ and $C_j$. The cosine similarity between these two entity term distributions becomes *term match* feature.

**Language Models**. The feature $LMScore$ is based on the language modeling approach to information retrieval [16]. We treat the card entity term set as the query and the user entity term set as the document. Then we compute the log likelihood of generating card entity terms using a language model constructed from user entity terms. We use Laplace smoothing in the computation of language model score.

**Word Embedding**. We extract semantic similarity features between entities based on word embeddings. Word embeddings [11,12] are continuous vector representations of words learned from very large data sets based on neural networks. The learned word vectors can capture the semantic similarity between words. In our experiment, we trained a Word2Vec model using the skip-gram algorithm with hierarchical softmax [12]. The training data was from the Wikipedia English dump obtained on June $6$th, $2015$. Our model outputs vectors of size $200$. The total number of distinct words is $1,425,833$. We then estimate entity vectors based on word vectors. For entities that are phrases, we compute the average vector of embedding of words within the entity phrase. After vector normalization, we use the dot product of entity vectors to measure entity similarity. To define features for the similarity of $U_i$ and $C_j$ , we consider feature variations inspired by hierarchical clustering algorithms as shown in table 2.

**Entity Taxonomy in Knowledge Base**. Another way to extract semantic similarity features between entities is measuring the similarity of entity taxonomy [10]. As presented in Section 4, we link entities in the user interest profile with entities in a large scale knowledge base. From the knowledge base, we can extract the entity taxonomy which is the entity type information. Two entities without any common terms could have similarities if they share some common entity types.

Table 3: Examples of entity taxonomy for “Kobe Bryant” and “Byron Scott”.

| Entity name | Kobe Bryant | Byron Scott |
|---|---|---|
| Taxonomy 1 | award.nominee | award.winner |
| Taxonomy 2 | award.winner | basketball.coach |
| Taxonomy 3 | basketball.player | basketball.player |
| Taxonomy 4 | celebrities.celebrity | event.agent |
| Taxonomy 5 | event.agent | sports.sports_team_coach |
| Taxonomy 6 | film.actor | film.actor |
| Taxonomy 7 | olympics.athlete | tv.personality |

Table 3 shows entity taxonomy examples for “Kobe Bryant” and “Byron Scott”. We can see that these two entities share common taxonomies like “basketball.player”, “award.winner”. They also have their own special taxonomies. “Kobe Bryant” has “olympics.athlete” in the taxonomies whereas “Byron Scott” has the taxonomy named “basketball.coach”. Based on this observation, we can measure the semantic similarity between two entities base on their taxonomies. Specifically, we measure the similarity of two entities based on the Jaccard index of the two corresponding taxonomy sets. Since all taxonomies only have two levels, we compute entity taxonomy similarity features in two different granularity. When we measure the similarity of the $U_i$ and $C_j$, we can compute the average similarity of all entity pairs in this two entity sets. We compute a weighted version where each entity is weighted by its impression count and a non-weighted version for this features. In summary, in this feature group, we have $4$ features that are listed in Table 2.

### 5.3 User Demographics Features (UD)

Part of user interests are influenced by their demographic information such as age and gender. The tastes of teenagers and adults are different. Men and women also have different preferences for information cards. Motivated by this intuition, we also extract features related to user demographic information. In addition to the raw user demographics features, we also add user demographics features in a *matched version*. We compute the matched features of the user demographics features between the user and users who clicked the card URLs. To achieve this, we need to compute the average age and average gender value for users that clicked on each card. The gender value is between $1$ (male) and $2$ (female), where the more the value is approaching $1$, the more men clicked the URLs in the corresponding card. Based on this, we compute the differences between user demographic features. We distinguish zero distance with null cases by adding an offset to zero distance when we compute the matched version feature values. The details of these features are explained in the Table 4.

Table 4: Summary of user demographics features (UD).

| Feature | Description |
|---|---|
| UserAge | Integer value of user's age. |
| UserGender | Binary value of user's gender. |
| UserLanguage | Integer value to denote user's language. |
| UserRegisterYears | Integer value to denote the number of years since user's registration. |
| CardAvgAge | Average age of all users who clicked the URLs on the card. |
| CardAvgGender | Average gender value of all users who clicked the URLs on the card. |
| AgeAbsDistance | The absolute distance for age between the user with all users who clicked card URLs. |
| AgeRelDistance | The relative distance for age between the user with all users who clicked card URLs. |
| GenderAbsDistance | The absolute distance for gender between the user with all users who clicked card URLs. |
| GenderRelDistance | The relative distance for gender between the user with all users who clicked card URLs. |

## 6 Experiments

### 6.1 Data Set and Experiment Settings

We use real data from a commercial intelligent personal assistant for the experiments. The training data is from one week between March 18th, 2015 and March 24th, 2015. The testing data is from one week between March 25th, 2015 and March 31st, 2015. The statistics of card impressions and users are shown in Table 5.

The user profiles represented by entities are built from multiple logs presented in Section 4. The time window from user profile is from March 18th, 2014 to March 17th, 2015. So there is no overlap time between the user profiles and training/testing data. Since most proactive impressions have only one card with a positive relevance label, we pick mean reciprocal rank(MRR) and NDCG@1 as the evaluation metric.

Table 5: Statistics of training data and testing data.

| Item | Training Data | Testing Data |
|---|---|---|
| # Cards | 8,499,640 | 9,400,779 |
| # Cards with URLs | 4,721,666(55.55%) | 4,920,380(52.34%) |
| # Cards with entities | 3,934,644(46.29%) | 3,960,484(42.13%) |
| # Users | 232,413 | 233,647 |
| # Users with entities | 210,139(90.42%) | 205,067(87.77%) |

### 6.2 Relevance Labels Generation

Following previous research in reactive search personalization [3,6] and proactive information card ranking[15], we use the SAT-Hybrid method to generate the relevance labels in our experiments. This method considers all cards with a *SAT-Click* or a *SAT-View* as relevant cards. The definition of *SAT-Click* and *SAT-View* are as following.

*SAT-Click*: For each card in proactive impressions, we consider clicked cards with $\geq$ 30s dwell time as relevant and other cards as non-relevant. This is a commonly used strategy for generating relevance labels in reactive search systems.

*SAT-View*: Some types of cards do not require a click to satisfy users' information needs. For instance, users could scan the weather and temperature information on the cards without any clicking behavior. Stock cards could also tell users the real-time stock price of a company directly in the card content. Cards with viewport duration $\geq$ 10s are labeled as relevant and the others are non-relevant.

### 6.3 Learning Models

We choose LambdaMART [20] as our learning model to rerank cards based on features extracted in Section 5. LambdaMART is an extension of LambdaRank [5]. This learning to rank method based on gradient boosted regression trees is one of the most effective models for the ranking task. It won Track 1 of the 2010 Yahoo! Learning to Rank Challenge and was commonly used in previous research on personalized ranking [3,4,18].

### 6.4 Comparison of Different Rankers

We compare the performance of different rankers. The baseline ranker is a production ranker which has been shipped to a commercial personal assistant system. This production ranker includes features that statically rank the different information cards based on their relative importance, and dynamic features that adjust their relevance scores based on the contextual information and the card content. This ranker is similar and comparable to the Carre model as described in [15]. We only report the relative gains and losses of other rankers against this production ranker to respect the proprietary nature of this ranker. The rankers which are compared with the baseline ranker include the following:

- UMPRanker-I(IF): The ranker from adding IF features on top of the features being used in the production ranker.
- UMPRanker-IE (IF + EF): The ranker from adding IF and EF features on top of the features being used in the production ranker.
- UMPRanker-IEU(IF + EF+ UD): The ranker from adding IF, EF and UD features on top of the features being used in the production ranker.

### 6.5 Experimental Results and Analysis

Table 6 summarizes the relative improvements of the different rankers against the baseline ranker. Starting from the base set of features used in the baseline model, we gradually add the three feature groups introduced in Section 5, namely, IF, EF and UD. IF is the feature group of directly modeling user historical interactions with the proactive cards, which is a coarser-level modeling, based on the card type, while EF is the finer-grained modeling at the level of entities. UD is the group of demographical features, which can be seen as a multiplier/conditioner on top of the first two feature groups for additional gains.

As we can see, with the IF features, we were able to capture the user interests nicely, resulting in significant improvements of 2.18% in MRR and 2.25% in NDCG@1 compared to the strong baseline ranker that was shipped to production, which is very substantial. On top of the baseline features and IF features, adding EF features, we were able to see significant larger gains of 2.37% in MRR and 2.38% in NDCG@1, demonstrating the substantial additional values in the entity-level modeling. Finally, with the UD features added, we were able to see additional statistically significant gains, even though to a lesser extent, making the total improvements of MRR to NDCG@1 both to 2.39%.

Table 6: Comparison of different rankers with the production ranker. The gains and losses are only reported in relative delta values to respect the proprietary nature of the baseline ranker. All differences are statistically significance ($p < 0.05$) according to the paired t-test.

| Method | $\Delta$MRR | $\Delta$NDCG@1 |
|---|---|---|
| UMPRanker-I (IF) | +2.18% | +2.25% |
| UMPRanker-IE (IF + EF) | +2.37% | +2.38% |
| UMPRanker-IEU (IF + EF+ UD) | +2.39% | +2.39% |

### 6.6 Feature Importance Analysis

Next we perform feature importance analysis. By analyzing the relative importance, we can gain insights into the importance of each feature for the proactive ranking task. LambdaMART enables us to report the relative feature importance of each feature. Table 7 shows the top 10 features ordered by feature importance among IF, EF and UD feature groups. Half of the most important 10 features come from the IF feature group. 3 features come from the EF feature group and the rest are from the UD feature group. Features with the highest feature importance are $ViewRate$, $KBTaxonomyLevel1Weighted$, $CTR$, $AverageViewTime$ and $LMScore$. Features like $ViewRate$, $CTR$, $AverageViewTime$ can capture users' preferences on different card types based on user historical interaction with the intelligent assistant system. Entity based features like $KBTaxonomyLevel1Weighted$ and $LMScore$ are useful for improving proactive ranking through modeling user interests with user engaged textual content with term matching and semantic features. UD features, as shown in Table 7, are not as important as IF and EF features. However, they can still contribute to a better proactive ranking by capturing user preferences with user demographics information.

Table 7: The most important features learnt by LambdaMART.

| Feature Name | Feature Group | Feature Importance |
|---|---|---|
| ViewRate | IF | 1.0000 |
| KBTaxonomyLevel1Weighted | EF | 0.9053 |
| CTR | IF | 0.8593 |
| AverageViewTime | IF | 0.7482 |
| LMScore | EF | 0.6788 |
| ViewRate | IF | 0.4948 |
| CardAvgAge | UD | 0.1026 |
| SATCTR | IF | 0.0705 |
| TMWeighted | EF | 0.0628 |
| GenderAbsDistance | UD | 0.0486 |

### 6.7 Case Studies of Re-ranking

To better understand the improvements in ranking enabled through our UMPRanker, we conduct case studies to look into the changes in the re-rankings of the proactive cards. From the examples, we find that the UMPRanker is able to identify the individual card types that each user prefers and rank them higher for the user (e.g., for users who like restaurant cards, the cards are promoted higher), thanks to the IF features; and provide customized ranking for different demographics, thanks to UD features (e.g., promoting sports cards for male users). And finally, we also observe the proposed EF features allow finer-grained improvements to adapt the ranking according to the user interests at the entity-level. Table 8 provides an example of this. As we can see, two News cards were promoted (i.e., News1 from 3rd to 1st, News2 from 4th to 2nd), while one News card (i.e., News3 from 2nd to 4th) was demoted. A closer look at the data reveals that the two promoted news cards are of higher weights learned in the EF representations of the user interests due to higher historical engagements with the entities (embedded in the news articles of News1 and News2) for the user, showing the benefits of finer-grained modeling such as EF on top of the coarser-grained user interests modeling at the card type level through IF.

Table 8: Examples of reranked cards in the testing data. “IsSuccess” denotes the inferred relevance labels based on SAT-Click or SAT-View with the timestamp denoted by “Time”.

| CardType | RankBefore | RankAfter | IsSuccess | Time |
|---|---|---|---|---|
| News1 | 3 | 1 | TRUE | (3/28 8:11) |
| News2 | 4 | 2 | TRUE | (3/28 8:13) |
| Calendar | 1 | 3 | FALSE | |
| News3 | 2 | 4 | FALSE | |
| News4 | 5 | 5 | FALSE | |
| Restaurant | 7 | 6 | FALSE | |
| Places1 | 8 | 7 | FALSE | |
| Sports | 10 | 8 | FALSE | |
| Places2 | 9 | 9 | FALSE | |
| Weather | 6 | 10 | FALSE | |

## 7 Conclusion and Future Work

In this paper, we explore a variety of ways to model user interests, with the focus on improving the ranking of information cards for proactive systems such as Google Now and Microsoft Cortana. We propose a learning to rank framework and encode the various models as features, which include coarser-grained modeling of card type preferences directly mined from the historical interactions, finer-grained modeling of entity preferences, and features that capture the variations among demographics. Experiments performed with large-scale logs from a commercial proactive search system show that our method significantly outperforms a strong baseline method deployed in production, and show that the fine-grained modeling at the entity-level and demographics enable additional improvements on top of the coarser-grained card-type level modeling. In the future, we plan to experiment with different strategies, such as collaborative filtering, to further address the feature sparsity in entity-level modeling and contextualize the user interest modeling on factors such as time and location.

## 8 Acknowledgments

This work was done during Liu Yang’s internship at Microsoft Research and Bing. It was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-1419693. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor. We thank Jing Jiang and Jiepu Jiang for their valuable and constructive comments on this work.

## References

1. E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06*, pages 19–26, New York, NY, USA, 2006. ACM.
2. J. Allan, B. Croft, A. Moffat, and M. Sanderson. Frontiers, challenges, and opportunities for information retrieval: Report from swirl 2012 the second strategic workshop on information retrieval in lorne. *SIGIR Forum*, 46(1):2–32, May 2012.
3. P. N. Bennett, F. Radlinski, R. White, and E. Yilmaz. Inferring and using location metadata to personalize web search. In *SIGIR '11*, July 2011.
4. P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short- and long-term behavior on search personalization. In *SIGIR '12*, pages 185–194, New York, NY, USA, 2012. ACM.
5. C. Burges, R. Ragno, and Q. Le. Learning to rank with non-smooth cost functions. In *NIPS '07*. MIT Press, Cambridge, MA, January 2007.
6. S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst.*, 23(2):147–168, Apr. 2005.
7. Q. Guo, H. Jin, D. Lagun, S. Yuan, and E. Agichtein. Mining touch interaction data on mobile devices to predict web search result relevance. In *SIGIR '13*, pages 153–162, 2013.
8. D. Lagun, C.-H. Hsieh, D. Webster, and V. Navalpakkam. Towards better measurement of attention and satisfaction in mobile search. In *SIGIR '14*, pages 113–122, New York, NY, USA, 2014. ACM.
9. J. Li, S. Huffman, and A. Tokuda. Good abandonment in mobile and pc internet search. In *SIGIR '09*, pages 43–50, New York, NY, USA, 2009. ACM.
10. T. Lin, Mausam, and O. Etzioni. No noun phrase left behind: Detecting and typing unlinkable entities. In *EMNLP-CoNLL '12*, pages 893–903, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.
11. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
12. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS'13*, pages 3111–3119, 2013.
13. P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, Mar. 1997.
14. B. J. Rhodes and P. Maes. Just-in-time information retrieval agents. *IBM Syst. J.*, 39(3-4):685–704, July 2000.
15. M. Shokouhi and Q. Guo. From queries to cards: Re-ranking proactive card recommendations based on reactive search history. In *SIGIR'15*, May 2015.
16. F. Song and W. B. Croft. A general language model for information retrieval. In *CIKM '99*, pages 316–321, New York, NY, USA, 1999. ACM.
17. X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, Jan. 2009.
18. L. Wang, P. N. Bennett, and K. Collins-Thompson. Robust ranking models via risk-sensitive optimization. In *SIGIR '12*, pages 761–770, New York, NY, USA, 2012. ACM.
19. R. W. White, W. Chu, A. Hassan, X. He, Y. Song, and H. Wang. Enhancing personalized search by mining and modeling task behavior. In *WWW '13*, pages 1411–1420, Republic and Canton of Geneva, Switzerland, 2013.
20. Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Inf. Retr.*, 13(3):254–270, June 2010.
21. S. Xu, H. Jiang, and F. C.-M. Lau. Mining user dwell time for personalized web search re-ranking. In *IJCAI'11*, pages 2367–2372, 2011.