Predicting the Topic of Your Next Query for Just-In-Time IR

Seyed Ali Bahrainian[1], Fattane Zarrinkalam[2], Ida Mele[3], and Fabio Crestani[1]

[1] Faculty of Informatics, University of Lugano (USI), Switzerland,
[2] Laboratory for Systems, Software and Semantics (LS^3), Ryerson University,
[3] ISTI-CNR, Pisa, Italy
{bahres, fabio.cerstani}@usi.ch, fzarrinkalam@ryerson.ca, ida.mele@isti.cnr.it

Abstract. Proactive search technologies aim at modeling the users' information seeking behaviors for a *just-in-time information retrieval* (JITIR) and to address the information needs of users even before they ask. Modern virtual personal assistants, such as Microsoft Cortana and Google Now, are moving towards utilizing various signals from users' search history to model the users and to identify their short-term as well as long-term future searches. As a result, they are able to recommend relevant pieces of information to the users at just the right time and even before they explicitly ask (e.g., before submitting a query). In this paper, we propose a novel neural model for JITIR which tracks the users' search behavior over time in order to anticipate the future search topics. Such technology can be employed as part of a personal assistant for enabling the *proactive* retrieval of information. Our experimental results on real-world data from a commercial search engine indicate that our model outperforms several important baselines in terms of predictive power, measuring those topics that will be of interest in the near-future. Moreover, our proposed model is capable of not only predicting the near-future topics of interest but also predicting an approximate time of the day when a user would be interested in a given search topic.

Keywords: Topic Prediction, Topic Modeling, Just-In-Time Information Retrieval, Neural IR

1 Introduction

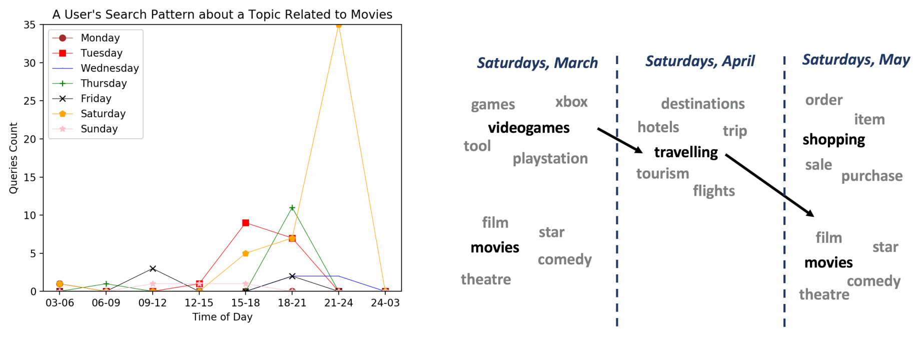
With the rapid proliferation of web search as the primary mean for addressing the users' information needs, search engines are becoming more sophisticated with the purpose of improving the user experience and of assisting users in their search tasks more effectively. As an example, with the increasing and ubiquitous usage of mobile devices, it has become more important for search engines to offer also *Just-In-Time Information Retrieval* (JITIR) [18] experiences. This means retrieving the right information at just the right time [10] to save users from the hassle of typing queries on mobile devices [2, 3].

The notion of "personalized search" [25] has shown to be effective in improving the ranking of search results. However, such personalization comes at the cost of lower speed, which in some cases might even cause the retrieval of the results only after the user search session has ended. Moreover, possible discovery of newly available content related to a previous search is another application of JITIR models for presenting results to a user at a future time.

As a result, researchers have focused on improving search personalization with respect to not only the retrieved content but also the user's habits (e.g., *when and what* information the users consume). While such models can benefit desktop users in better addressing their information needs at just the right time, they are essential on mobile platforms. Indeed, Microsoft Cortana and Google Now aim at offering a proactive experience to the users showing *the right information before they ask* [24].

As pointed out by Agichtein et al. [1], knowing the user's information needs at a particular time of the day allows to improve the search results ranking. For example, the search results can be personalized based on the specific search task (of a given user at a given time) rather than based on the more general information of user interests which have been inferred by the entire user's profile. This would also support users in resuming unfinished search tasks (e.g., if a search is likely to be continued one can save the results already found for a faster or more convenient access once the task is resumed).

Figure 1(a) shows the behavior of a randomly selected user from our dataset in issuing search queries related to a topic about `movies` over differnet week days. For example, the user might have searched the word "imdb" along with the title of a movie. As we can see, the user exhibits a higher tendency to search for movies in the afternoons and evenings as well as on Saturdays. Hence, we can infer that the user is interested in watching movies on Saturday evenings and thus it is likely that her queries are related to movies. Moreover, as shown in Figure 1(b) a user changes search behavior over time. To address such changes in search behavior we propose a dynamic memory system.



(a) (b)

Fig. 1. (a) The number of queries about `movies` submitted by a randomly selected user. (b) Evolution of user-search patterns on different Saturdays.

Addressing the near-future information needs of the users has been also studied in the context of personal assistants, such as Google Now, Microsoft Cortana, or Apple's Siri and in the context of memory augmentation in meetings [6]. These systems offer proactive experiences [22] that aim to recommend useful information to a user at just the right time.

In this paper, we focus on predicting the topics of the users' future search queries. Specifically, we propose a model which predicts the topic of the search queries submitted by the users in the next 24 hours. Moreover, our model leverages the user's behavior patterns over time in order to predict the topic of the user's query on a specific weekday (e.g., Mondays, Tuesdays) and at an approximate time of the day. The main contributions of this paper are:

C1: we propose a time-series model based on neural networks to predict the topic of near-future queries of users.

C2: our model is equipped with a dynamic memory learning users' behavior over time. This memory evolves over time when the search patterns change. We demonstrate that our dynamic memory architecture is beneficial as it increases the prediction performance. Further, we believe that this model could be useful in other domains that involve temporal data.

The organization of this paper is as follows: Section 2 presents the related work, Section 3 describes our research goals and Section 4 presents our model for predicting the topics of the users future search queries. In Section 5, we evaluate our method against the baseline methods based on their predictive performance. Finally, Section 6 concludes this paper and gives insight into future work.

2 Related Work

2.1 Just-In-Time Information Retrieval

Addressing the users' near-future information needs has been studied in the context of personal assistants [4] such as Google Now, Microsoft Cortana, or Apple's Siri. These systems offer proactive experiences and aim to recommend the right information at just the right time [22,24]. As an example, Sun et al. [24] proposed an approach for tracking the context and intent of a user leveraging smartphone data [5] in order to discover complex co-occurring and sequential correlations between different signals. Then, they utilized the discovered patterns to predict the users' intents and to address their information needs.

In the context of proactive search and recommendation, Song and Guo [23] aimed at predicting task repetition for offering a proactive search experience. They focused on predicting when and what type of tasks will be repeated by the users in the future. Their model was based on time series and classification. They tested the effectiveness of their approach for future query and future app predictions. Our work differs from their work since we take a collaborative time-series approach for predicting the topics of future user queries. Moreover, our goal is to predict the topic of one's next query and not only predicting the repetition of a search task.

Agichtein et al. [1] tried to predict the continuation of a previously started task within the next few days. Similarly to [23], they defined the prediction of the continuation of a task as a classification problem. They used an extensive set of features for training the classifiers. Such features include query topics, level of user engagement and focus, user profile features such as total number of unique topics in prior history, and repeating behavior among others. Our work differs from this work as we do not simply try to predict the search task continuation in the future but we also aim at predicting the day of the week and the approximate time of the day when a query topic will occur. Moreover, unlike their model which is a classifier based on a number of hand-engineered features, our model has a time-series structure and it evolves over time by learning from the data and correcting itself over time.

Furthermore, another interesting but different work consists in the identification of recurrent event queries and was presented by Zhang et al. [28]. In this work, the authors aimed at identifying search queries that occur at regular and predictable time intervals. To accomplish this, they train various classifiers such as Support Vector Machines and Naïve Bayes, on a number of proposed features such as query frequency, click information, and auto correlation. They conclude that a combination of all features leads to the highest performance.

2.2 Topic models and Word Embeddings

Topic models are defined as hierarchical Bayesian models of discrete data, where each topic is a set of words, drawn from a fixed vocabulary, which together represent a high level concept [26]. According to this definition, Blei et al. introduced Latent Dirichlet Allocation (LDA) [8]. In our work, we use LDA for discovering the topics of the users' search queries. In particular, as we will see in Section 4, we created a collection of documents consisting of some of the query results, then we run LDA to extract the topics of the various search queries.

Another form of word vectors is represented by word embeddings which map semantically related words to nearby positions in a vector space. Topic modeling approach is also unsupervised. Some well-known approaches are the word2vec model [15] and the Glove model [17]. As explained in Section 4, we needed to use word embeddings to model the dependencies between different attributes.

3 Research Goals

We can summarize the goals of our work as follows: (1) predicting the topics of future search queries of a user, and (2) predicting the day of the week and the approximate time of the day when the topic will be queried by a user.

Given the search history of each user u in the last n consecutive time slices, as well as a set of corresponding query topics $\mathbb{Z}$, we aim to predict the topic $z \in \mathbb{Z}$ of the query of user u in the $(n+1)_{th}$ time slice.

For achieving this, we first model the search tasks as topics using LDA. Then, leveraging a time-series model we discover the latent patterns in search tasks and predict the continuation of a search task in the near-future. In other words, we aim at predicting the topics of the user's future queries. Such technology will enable the proactive retrieval of relevant information in a JITIR setting. However, estimating the time of the day when a user would access a particular content is the second piece of the puzzle in order to recommend content more precisely and more effectively. Thus, our second goal consists in correctly predicting when (day and time of the day) the users will consume what content (topic) knowing the users' habits in requesting the various topics at the different times.

4 Query Topic Prediction Model

We now present our novel time-series evolutionary model for predicting the topic of a user's near-future queries. The model is based on the notion of reinforcement learning so that it adapts itself to the data over time and corrects itself. We formally define our model as a function f which takes as input the search history of users and predicts which topics occur in the near future.

The model consists of a dynamic memory in the form of a word embedding connected with a *Bi-directional Long Short Term Memory* (BiLSTM) [21] used to capture the behaviour of a user over time. The dynamic memory implements two different effects of persistence and recency. At each point in time, based on the possible changes in the input data, it updates the word vectors to provide as input to the BiLSTM network.

In the following, we first describe the dynamic memory system in Sections 4.1 and 4.2. Then, we present the BiLSTM network in Section 4.3.

4.1 A Dynamic Memory based on Word Embeddings

Our intuition behind the design of such memory model is that people often show similar behavior over time (i.e., persistence) but they also have a tendency to explore new things (i.e., recency). As a result, over a timeline people may show very different behaviors and the model should be capable of capturing them in order to accurately anticipate the users' future behaviors [27]. Therefore, we believe that dividing the temporal input data into a number of time slices and weighting them based on identified patterns in the data is important.

The dynamic memory is based on the word2vec word embeddings. Throughout this paper whenever we use the term word2vec, we refer to a Skip-Gram with Negative Sampling (SGNS) word embedding model. Levy et al. [14] showed that the SGNS method is implicitly factorizing a word-context matrix, whose cells are the Pointwise Mutual Information (PMI) of the respective word and context pairs, shifted by a global constant. They further elaborate that word2vec decomposes the data very similar to Singular Value Decomposition (SVD) and that under certain conditions an SVD can achieve solutions very similar to SGNS when computing word similarity. Apart from scalability and speed, SGNS is capable of removing bias towards rare words using negative sampling. Other than the few differences, at the concept level both SVD and SGNS are very similar. They both build a word-context matrix for finding similarities between words.

Based on these principles we propose a novel and effective method for integrating multiple word2vec memory components where each is trained with data from a different time slice of the input data. Let $m_t \in \boldsymbol{M}$ where m_t is a word2vec memory trained on data form time slice t and $\boldsymbol{M}$ is a vector of all word2vec models. Instead of using only one single memory to capture the global patterns in the dataset, we propose to use a different word vector from model m_t to represent time slice t where $t \in 0, 1, \ldots, n$. Then, we integrate all these word vectors into one final vector. Therefore, a temporal dataset of web search queries can be divided into n different time slices, and one word2vec memory m_t is trained for each time slice. We assume that all the vectors have the same embedding dimensions, so given two vectors m_t and m_{t+1} we can combine them using the *orthogonal Procrustes* matrix approximation. Let W^t be the matrix of word embeddings from m_t which is trained on data at the time slice t. We align across m_t and m_{t+1} which are derived from consecutive time slices while preserving the cosine similarities by optimizing:

$$argmin_{Q^T Q = I} ||W^t Q - W^{t+1}|| \tag{1}$$

Matrix Q is described in the following. We note that this process only uses orthogonal transformations like rotations and reflections. We have solved this optimization problem by applying SVD [20].
We can summarize the steps of the approach as follows:

1. The vocabulary of the resulting word vectors from the two time slices are intersected and the ones in common are kept. We note that due to our definition of an active user as well as the way we map queries to unique user, topic and time identifiers, vocabulary remains the same over all time slices (see Section 5.1).
2. We compute the dot product of the two matrices (for doing so, we first transpose one of the matrices).
3. The SVD of the matrix resulting from the dot product is computed. The result consists of three factorized matrices commonly known as U, the diagonal matrix S, and the matrix V.
4. We compute the dot product of U (left singular matrix) and V (right singular matrix) to have as resulting matrix Q. Since S contains information on the strength of concepts representing word-dimension relations which are not needed here as they are not modeled in word2vec, we discard the matrix S. The existence of the S matrix is also one important difference between SVD and word2vec, which word2vec does not compute.
5. Finally, we compute the dot product of Q and the embedding matrix W^t. For further detailed information we refer to [20] where the *orthogonal Procrustes* approximation using SVD is described.

We repeat the process of model alignment for all n word2vec models spread over the entire timeline.

4.2 Modifying the Dynamic Memory using Recency and Persistence Effects

Now that we have explained the process of combining different word2vec models, in this section we explain how our proposed model takes into account the recency and persistence of the searching behaviors of users over time. Before combining W^t and W^{t+1} which are word-dimension matrices from two word2vec models (as described in Section 4.1), we modify each matrix based on the following effects:
Recency Effect. It modifies the strength of word embeddings by assigning higher weights to the word vectors observed in the most recent time slice. We formally define the recency effect as follows: given the query topics of the last n consecutive time slices of a sequential dataset, we would like to predict which query topics continue in the $(n+1)_{th}$ time slice. By assuming a vocabulary v of all the words occurring in the first n time slices, we construct a word vector containing the probability scores corresponding to each word in v. The assigned probability scores are higher for the words appearing in the most recent time slices. After modifying the word vectors, we then perform alignment of models as described in Section 4.1. According to the recency effect presented in the following equation we modify the word embedding matrices W^ts by $P_{Rec} = \sum_{n=1}^{N} \sum_{w_i \in W^t} P(w_i) * 2^n$, where n is the time slice number, P indicates probability, and w_i is a word from the word embedding matrix W^t. The 2^n is the rate with which recent word vectors are assigned higher weights. The resulting constructed word vector is an average representation of the probability of all the words present in all the n time slices.

Therefore, this effect assigns higher weight to a word which has occurred in the most recent time slice of a sequential corpus. We refer to the word vectors which are computed by the recency effect as the *recency matrix*.
Persistence Effect. Given the word embeddings of the last n consecutive time slices, we would like to predict which query topic continues in the $(n+1)_{th}$ time slice. Given a vocabulary v of all the words occurring in the first n time slices, we construct a word vector containing the probability scores corresponding to each word in v. The assigned probability scores are higher for the words which have persisted over time. We compute the updated probability of each word according to the persistence effect using $P_{Pers} = \sum_{n=1}^{N} \sum_{w_i \in W^t} P(w_i) * 2^{-n}$, where n is the time slice number, P indicates probability, and w_i is a word from W^t. The 2^{-n} is the rate with which the higher weights are assigned to persistent words. Therefore, the more persistent words (i.e., persisting in occurrence) have higher weights, and we refer to the word vector computed with the persistent effect as the *persistence vector*.
Combining Recency and Persistence: Recency and persistence scores are combined in a linear interpolation to modify the original word embedding matrix. The linear interpolation at the time t is defined as:

$$EmbeddingMatrix_{w,t} = w_{P,t} * Score_{Pers.} + w_{R,t} * Score_{Rec.} \tag{2}$$

where $Score_{Pers.}$ and $Score_{Rec.}$ are computed by the persistence and the recency effects, respectively. Furthermore, $w_{P,t}$ and $w_{R,t}$ are persistence weights and recency weights computed at each time slice. They have the following relation and are learned from the data: $w_{P,t} + w_{R,t} = 1$. This means that at each time slice t each of the two effects corresponds to a weight. The weights can be equal (i.e. when the effects have the same intensity) or different, but their sum would be always 1.

The weight $w_{R,t}$ is then computed as square root of the sum of the difference in the number of occurrences of each query topic compared with the previous time slice divided by the number of all the queries at the same time slice. Subsequently, $w_{P,t}$ is computed based on $w_{P,t} + w_{R,t} = 1$. As a result, the dynamic memory evolves over time and updates itself proportionally to the rate of the changes in the data.

Finally, we map each query to a topic using LDA. Further details of this process are explained in Section 5.1. We specify each query with the ID of the user who submitted it along with the given week day and time bucket (i.e. which is an approximate time of day) of the query. Then, we train n word2vec SGNS models on this data in order to train the dynamic memory. For word2vec, we use embedding size of 300, without discarding any of the input words. The result will be n word embedding matrices derived from n time slices which are aligned and combined into one word embedding matrix which is given as input to the BiLSTM.

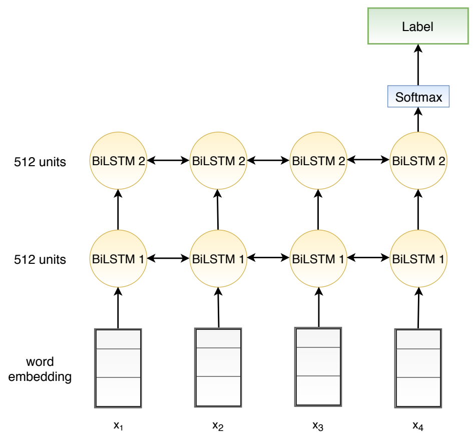


Fig. 2. The architecture of our proposed model. x_i stands for input at time step i

4.3 Bi-directional Long Short Term Memory (BiLSTM)

We train the BiLSTM network using the word embeddings of the dynamic memory. We use the BiLSTM neural network as function for generating a sequence of events given an input query. In other words, we aim at modeling the sequence of observations (i.e., the searches about certain topics) in a time-series fashion. Thus, given the user ID, the future week day and the time bucket, the model will predict the topics of the near-future queries.

As shown in Figure 2, the architecture of our model consists of word embeddings from the dynamic memory provided as input to the BiLSTM network. We model each query in 4 recurrent time steps in order to predict the topic of near-future queries along with their weekday, and approximate time of the day (i.e., we refer to it as the time bucket in dataset description). On the other hand, when we want to only predict the near-future topics without specifying their approximate time of the day, we train the same network with 2 recurrent time steps (i.e., one for the user ID and the other for the topic). In both cases we set number of word2vec models to six (n=6) to model almost every two weeks with one word2vec model. Furthermore, our model includes two fully connected BiLSTM layers, with each layer containing 512 cells or units. We applied a SoftMax layer to the final output from the BiLSTM networks.

Our intuition behind this architecture is to first find a collaborative generalization of patterns of users in issuing queries about certain topics at particular points in time by using the dynamic memory based on the word2vec model. Then, using the BiLSTM neural network we leverage the local dependencies between certain behaviors in a temporal manner. The BiLSTM network serves as a time-series model that determines the occurrence of a future event (i.e., a future query's topic) by modeling the sequences of events (i.e., sequences of topics).

5 Experimental Setup

5.1 Dataset Description

In the experiments, we use the publicly available AOL query log [16] which has been used in other research works on query caching and search result personalization. It consists of about 36M query records and spans a period of three months (from March to May 2006). Each record consists of the anonymous ID of the user, query keywords, timestamp, and rank and URL of the clicked result.

Our goal is to predict the topics of the future queries issues by a user, hence we selected those users who have a high number of queries. Formally, we define *active users* those who have searched at least one query every week and over a span of three months have issued at least $1,000$ queries. From this set which contains $1,197$ active users, we randomly selected 500 users to train and test our proposed model as well as the baselines. The query log made of the queries issued by these 500 users consists of $755,966$ queries.

Training and testing data. Our experiments aim at predicting the topics of the future queries searched by a user, so we sorted the query log by time and split it into training and test sets. The training set is used for learning the topics of interests of a user, while the test set to check the prediction performance. For our experiments, the test set consists of the queries issued in the last 24 hours (which results of $10,848$ queries) while the rest of the queries is used for training.

Modeling search tasks as topics. In order to model the topics of the search tasks we used the Latent Dirichlet Allocation (LDA) topic model [8].

Since the search queries are short and lack context, we decided to enrich them with the content of clicked pages. More in detail, given the queries from the training set and the URL of their clicked results, we gathered the content of 351, 301 unique web pages. We treat each query and the text of its corresponding clicked result as a document, and we run LDA over the collection made of these documents. LDA returns K list of keywords representing the latent topics discussed in the collection. Since the number of topics (K) is an unknown variable in the LDA model, it is important to estimate it. For this purpose, similar to the method proposed in [7, 9], we went through a model selection process. It consists in keeping the LDA parameters (commonly known as α and η) fixed, while assigning several values to K and run the LDA model each time. We picked the model that minimize $logP(W|K)$, where W contains all the words in the vocabulary. This process is repeated until we have an optimal number of topics. The training of each LDA model takes nearly a day, so we could only repeat it for a limited number of K values. In particular, we trained the LDA model with K equals to 50 up to 500 at steps of 50, and the optimal value was 150.

Labels for Predicting the approximate time. The search queries have timestamps, so we could extract the day of the week and the time of the day when they were issued. We divide the 24 hours into 8 time buckets of 3 hours each. Each time bucket represents an approximate time of the day and we can use this for predicting the approximate time of the day when a query topic will appear. Hence, given a user, our ultimate purpose is to predict the right query topic and when it will be requested (i.e., the week day and the time bucket).

5.2 Evaluation Metrics and Baselines

Evaluation Metrics. We performed a rigorous testing of our proposed method and compared it against several baseline methods. For our evaluation, we used the standard information retrieval evaluation metrics: precision, recall, and F_1.

Baseline Methods. Since our proposed method is based on a collaborative filtering principle, we chose as baselines the following top-performing techniques:

1. ***Probabilistic Matrix Factorization (PMF)*** is a model for collaborative filtering that has achieved robust and strong results in rating prediction [19].
2. ***Non-negative Matrix Factorization (NMF)*** can analyze data with a high number of attributes [13]. It reduces the dimensions of data by converting the original matrix into two matrices with non-negative values.
3. ***User-based K Nearest Neighbours (userKNN)*** is another popular method which uses similarities among the users' behaviours to recommend items to users.
4. ***SVD++*** is a collaborating filtering method, where previously seen patterns are analyzed in order to establish connections between users and items [11]. The approach merges latent factor models that profile both the users and the items with the neighborhood models that analyze the similarities between the items or between the users.

5. ***TimeSVD++*** is one of the most successful models for capturing the temporal dynamics and has shown strong results in various prediction and ranking tasks which seek to model a generalized pattern over time [12]. The regularization parameter and the factor size are selected using a grid search over $\lambda \in \{10^0, \ldots, 10^{-5}\}$ and $k \in \{20, 40, 80, 160\}$.
6. ***BiLSTM+w2v*** we also add as a baseline our own model with only one word2vec model trained as input (i.e., see n=1 in Table 2).

5.3 Experimental Results

First experiment. The aim of our first experiment is predicting the topics of the queries issued by a user in the next 24 hours. Table 1 reports the results of our approach compared to the baselines. We observe that our method outperforms all the baseline models in terms of predicting the topics of one's queries in the future 24 hours with statistically significant improvement. We averaged the prediction results over the 500 users of our sampled data. As a result of this experiment, we could observe that our model is superior in predicting the topics of future queries compared to the other collaborative-filtering baselines.

Our proposed model features incorporating some principles that we believe have caused the superiority of our model. First, the dynamic memory not only learns users' search behavior but also considers the temporal dimension when modeling data. Furthermore, our model uses the recency and persistence effects and adjust itself to the data by measuring the behavior of the data and subsequently updating itself when needed. None of the baseline models, despite being powerful models, can model such complexities.

Second experiment. In the second experiment, we would like to investigate whether or not running and combining different word2vec models can improve the performance compared to one trained word2vec model. In particular, we divided our input data into chunks (e.g., weeks) and trained several word2vec models over them. Then, we compared the performance of one word2vec model trained over the whole timeline against the performances achieved with different numbers of word2vec models. We started by only having one word2vec model up to 12 models (one for each week of our dataset).

Table 1. A comparison of our proposed method against the baselines.

	Precision	Recall	F_1
PMF (%)	12.53	31.03	19.78
NMF (%)	13.65	35.11	21.23
UserKNN (%)	14.20	38.06	22.09
SVD++ (%)	12.60	30.43	20.20
TimeSVD++ (%)	28.46	14.62	20.00
BiLSTM+w2v (%)	34.28	36.04	35.12
Our Model (%)	48.19	38.44	42.77
Our Model+time prediction (%)	26.23	34.41	29.77

Table 2. A comparison of the prediction performance varying the number of word2vec models in terms of F_1 (n=1 means one model trained over the whole dataset, etc.).

	n=1	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
24 hours prediction (%)	35.12	38.63	39.14	41.56	42.93	42.77	41.46	40.34	40.52	40.12
24 hours + time-bucket prediction (%)	28.64	29.32	29.57	29.83	30.21	29.77	29.40	29.43	29.21	29.06

The results of this experiment are reported in Table 2 and show that training the word2vec model over different time slices performs better than having only one word2vec model trained over the entire dataset. Moreover, we could observe that increasing the number of models allows to gain higher performance, however, after some point the performance plunges. We can conclude that training several models is better than one, but the number of models should be chosen depending on the application. We could observe that the best results can be achieved training the models with roughly two weeks of data.

Our research goal was to design an intuitive time-series method for modeling the user behavior, specifically regarding search queries. The broader vision and strategy that we tried to incorporate into the model was that the users have the tendency to repeat the behavior (e.g., searching about the same topic in a sequence), but they also have consistent behaviors (e.g., searching for the same topic every Saturday night). Hence, incorporating these two dimensions into our model helped to improve the prediction performance. The concept behind our model may also be used in a personal assistant environment for modeling other types of data, tracking the user behavior over time and providing the user with the right information just-in-time the user might need it. Envisaging that a system can correctly predict the topic of your near-future query more than 40% of times among all possible options (i.e., in this case 150 topics) while also predicting the time bucket when you will show interest in that topic and presenting relevant information or targeted ads to you even before you have started searching on that topic is a very interesting result.

6 Conclusions

In this paper, we addressed the problem of predicting topics of future queries for just-in-time IR. For this purpose, we proposed a novel method and compared it against six baseline methods which have been extensively used in the literature for temporal and non-temporal collaborative filtering. We showed through experimental results that our method, generalizing the users' behavior and modeling the temporal recurrent patterns, outperforms all the baselines. The developed method could be implemented as a part of a proactive search system that aids people in their every day lives.

One interesting future work would be adapting our method to other domains. For example, analyzing various data modalities gathered by current personal assistant tools such as Microsoft Cortana could be an interesting direction.

References

1. E. Agichtein, R. W. White, S. T. Dumais, and P. N. Bennet. Search, interrupted: Understanding and predicting search task continuation. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 315–324, 2012.
2. M. Aliannejadi, M. Harvey, L. Costa, M. Pointon, and F. Crestani. Understanding mobile search task relevance and user behaviour in context. *CHIIR 2019*, 2018.
3. M. Aliannejadi, H. Zamani, F. Crestani, and W. B. Croft. In situ and context-aware target apps selection for unified mobile search. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1383–1392, 2018.
4. S. A. Bahrainian and F. Crestani. Towards the next generation of personal assistants: Systems that know when you forget. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17, pages 169–176, 2017.
5. S. A. Bahrainian and F. Crestani. Tracking smartphone app usage for time-aware recommendation. In *Digital Libraries: Data, Information, and Knowledge for Digital Lives*, pages 161–172, 2017.
6. S. A. Bahrainian and F. Crestani. Augmentation of human memory: Anticipating topics that continue in the next meeting. In *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval*, CHIIR '18, pages 150–159, 2018.
7. S. A. Bahrainian, I. Mele, and F. Crestani. Predicting topics in scholarly papers. pages 16–28, 2018.
8. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.
9. T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 2004.
10. R. Guha, V. Gupta, V. Raghunathan, and R. Srikant. User modeling for a personal assistant. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 275–284, 2015.
11. Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.
12. Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 447–456, 2009.
13. D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.
14. O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.
15. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
16. G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems (InfoScale '06)*, New York, NY, USA, 2006. ACM.
17. J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation.

18. B. J. Rhodes. *Just-in-time information retrieval.* PhD thesis, Massachusetts Institute of Technology, 2000.
19. R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264, 2007.
20. P. H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.
21. M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
22. M. Shokouhi and Q. Guo. From queries to cards: Re-ranking proactive card recommendations based on reactive search history. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 695–704, 2015.
23. Y. Song and Q. Guo. Query-less: Predicting task repetition for nextgen proactive search and recommendation engines. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 543–553, 2016.
24. Y. Sun, N. J. Yuan, Y. Wang, X. Xie, K. McDonald, and R. Zhang. Contextual intent tracking for personal assistants. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 273–282, 2016.
25. J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 449–456, 2005.
26. C. Wang, D. Blei, and D. Heckerman. Continuous time dynamic topic models. *Proc. of UAI*, 2008.
27. F. Zarrinkalam, M. Kahani, and E. Bagheri. Mining user interests over active topics on social networks. *Information Processing and Management*, 54:339–357, 03 2018.
28. R. Zhang, Y. Konda, A. Dong, P. Kolari, Y. Chang, and Z. Zheng. Learning recurrent event queries for web search. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1129–1139, 2010.