Exploration of Semantic-aware Approach for Contextual Suggestion Using Knowledge from The Open Web

Yuan Wang[1,2⋆], Yongfeng Zhang[3], Yi Zhang[4], Xintong Zhang[4], Jie Liu[1,2], and Yalou Huang[1,2]

[1] College of Computer and Control Engineering, Nankai University, Tianjin, China, 300071
[2] College of Software, Nankai University, Tianjin, China, 300071
[3] State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science, Tsinghua University, Beijing, China, 100084
[4] School of Engineering, University of California, Santa Cruz, USA, 95060
{nkwangyuan, zhangyf07}@gmail.com, {yiz@soe., xzhang92@}ucsc.edu, {jliu, huangyl}@nankai.edu.cn

Abstract. This paper describes our group's first attempt on the Contextual Suggestion Track of the Twenty-fourth Text REtrieval Conference (TREC 2015). The task aims to provide recommendations on attractions for various kinds of users under different and complex contexts. TREC provides two ways to participate in the track: one is to create a web server that can respond to contextual related queries called "Live Experiment", the other is to submit run files that have all the responses to the released requests called "Batch Experiment". For Live Experiment, due to lack of training data, our approach sticks closely to the defined relevance judgement criteria and context knowledge. We take linear interpolation to combine a variety of factors and contextual related knowledge. For Batch Experiment, we further consider domain preference under user attributes, and take existing Machine Learning based methods in principle. We show that feature engineering is a vital part for attraction suggestions. We find that the performance of suggestions to the provided user profiles and contexts has been improved using domain preference analysis.

Keywords: Contextual Suggestion, User Modeling, Semantic Analysis, Domain Preference Analysis

1 Introduction

The TREC 2015 Contextual Suggestion task investigates search techniques for complex information needs that are highly dependent on context and user interests as suggested by the guidelines. The task is to develop a system that is able to give ranked suggestion (attractions) for a particular user out of user rating profiles with a particular context. Roughly speaking, given a user, the task focuses on traveling suggestions based on

⋆ This work was finished when the first two authors were visiting University of California, Santa Cruz.

two hypotheses. 1) A traveller prefers a suggestion that is appropriate to her/his profile based on the user's historical preferences. 2) A traveller prefers a suggestion that is highly related to the context that can be more than a city, also including a trip type, trip duration, the type of group the person is traveling with or the season in which the trip will occur. So we need to model user preferences out of her/his profile and the context as well.

The majority of the systems presented in the past collect attractions from well-known location based recommender services[1–4] such as Google Places[5], Yelp[6], TripAdvisor[7], Foursquare[8] and etc. Most of them[1, 2, 4] exploited the open web to perform useful information extraction and collection. For example, Yang, et al[2] queried Yelp to gather reviews and ratings about attractions to build user opinion profiles, and then ranked attractions based on relevance between user profiles and attractions. Some of them[3] defined the personalized input to directly query these recommender service systems to get suggestions. Besides information gathering, participants tried many kinds of machine learning techniques, such as Learning To Rank methods[2, 5], Support Vector Regression[4], Hierarchical K-Means Clustering[6]. Vector Space Model (VSM) is widely used in contextual suggestions as well[1, 4, 7]. Thus, we observe one vital trend via the past works: feature extractions and machine learning techniques have been generally accepted to become the effective ways for the task.

Different from the track in years before, TREC provided an ad-hoc collection for suggestion this year, which consists of a set of 1,234,842 attractions. This becomes challenging due to the quantity and the variety of attractions. TREC provides two ways to participate, one is to create a web server that can respond to contextual related queries, called "Live Experiment", the other is to submit run files that include responses to all the requests, called "Batch Experiment". As to "Live Experiment", only one user is seen by our server each time, making it difficult to train sophisticated ranking models. This makes practical sense in real applications where we always face the same issue. To understand details of attractions, we have crawled a variety of indirect information about attractions from the open web, such as reviews, ratings and web page contents. Based on useful information gathering, we model user preference with multi-field descriptions in multiple text semantic modeling ways. We make effort to go the extra mile to build a contextual related dictionary to improve performance on the cases with more pieces about the trip. At last, we take linear interpolation to combine a variety of factors, as well as contextual related background knowledge from people and personal tags from a user. As for "Batch Experiment", a file with a set of requests that were made during the live experiment is available which provides us a chance to test machine learning methods during this phase. We model the suggestion task as a classification problem, where we predict the rating of each triplet of "user, attraction, context" ranging in (0, 1, 2, 3, 4), and then rank candidates according to the probability of classification results. Here, we want to investigate user domain preference and further extend our model with domain preference under user's attributes such as gender and age, taking advantage of

[5] https://www.google.com/business/
[6] http://www.yelp.com/
[7] http://www.tripadvisor.com/
[8] https://foursquare.com/

useful information gathered in Live Experiment as well. After feature generation, we train prediction models by using Weka[8], and submit two runs based on methods that show the best performance with 5-fold cross-validation on training data sets.

The remainder of the paper is organized as follows. Section 2 describes some preliminary knowledge about the task. In Section 3, we show how to obtain the used data. In Section 4 and Section 5, we describe the process of feature extraction and experimental setup for Live Experiment and Batch Experiment. The results are shown in Section 6. We concludes the paper in Section 7.

2 Task Formulation

This section describes some preliminary information about the task. First, we describe the input and output of the task. Second, we discuss the evaluation metrics used to estimate the performance of different solutions.

2.1 Task Description

The task is to develop a system that is able to generate a ranked list of up to 50 suggested attractions for a particular person (based upon her/his profile) with a particular context.

TREC provides the complete set of contexts, an ad-hoc collection that consists of a set of attractions and user profiles. Each profile corresponds to a single user, and indicates trip-related context and user's preference with respect to each example suggestion. Each user indicates her/his preference with a rating from 0-4 for an attraction. If the user assigns a score equal to or greater than 3, it means that the user likes this suggestion. Otherwise, if the user assigns a score equal to or less than 1, it means that the user dislikes this suggestion. In the attraction collection file, there are an attraction ID, a context (city) ID, a URL and a title for each attraction. Note that this title is not always shown as the title of the page to that URL (contained in the "title" element under the HTML document's head). Each context is a city in which the user is located, which consists of an ID, a city name, a state name and an approximate latitude and longitude. Additionally, we can find more pieces of optional data about the trip, such as the trip type, the trip duration, the type of group and the season the trip will occur in.

2.2 Evaluation Metrics

The TREC 2015 Contextual Suggestion Track takes Precision at Rank 5 (P@5) as main metrics. The track also uses Mean Reciprocal Rank (MRR) to evaluate the final results.

3 Data Cleaning and Useful Information Gathering

In this section, we describe how we create a sub-collection from the published attraction collection and gather useful information. Due to so many noisy URLs in the raw data, we manually make some rules to select reliable URLs. We pick up two types of URLs as suggested candidates. One is the URLs of the attraction pages from well-known social networks, namely Yelp, Foursquare and TripAdvisor, the other is the URLs of home pages.

3.1 Attractions from Yelp, Foursquare and TripAdvisor

Many commercial online review websites such as Yelp, Foursquare and TripAdvisor provide friendly APIs for developers to access their content, such as business information, ratings and reviews. We detect the URLs with more information about attractions on these websites as **Tourist Sub-collection**. We keep the URLs that start with one of the following parts:"https://foursquare.com/v/", "http://www.yelp.com/biz/" or "http://www.tripadvisor.com/Attraction". For the attractions of these URLs, we query their corresponding API to get more information about attractions, such as their titles, web page contents, positive reviews, negative reviews, all ratings and their average ratings. Due to the fact that the content of a page contains so much useless information, we use TextRank[9] to learn the review summary about the attractions in place of the full page content as an additional field. Thus, we obtain the URL, the title, ratings, positive and negative reviews, the review summary for each attraction.

3.2 Attractions from Home Page

The home page of an attraction is expected to contain the most important and valuable information about the attraction. Thus, we extract all URLs that include equal to or less than 3 forward slash "/" as **Homepage Sub-collection**. Due to homepages shown without reviews or ratings, we query Yelp API[9] using the location and the title of the home page. In this step, the attraction whose location and title are the same as the input will be returned and crawled for more information. We summarize content of the home page to get summaries of attractions in the Homepage Sub-collection. This is the difference of the summary process between the Tourist Sub-collection and the Homepage Sub-collection.

All the crawlers are developed by Python and data is stored in json format. We have crawled 155,116 attractions in the Homepage Sub-collection and 479,058 attractions in the Tourist Sub-collection. We map the numeric rating scale of 1-5 or 1-10 into 1-5 scale. In sub-collections, we discard the ones of which average ratings are less than 3 to obtain general good attractions. The attractions from the Homepage Sub-collection and the Tourist Sub-collection are used to extract features for each candidate attraction and to model user preference.

4 Live Experiment

In this section, we describe our approach for generating a personalized rank list of suggested attractions for each "user, context" pair in Live Experiment. First, we present our way of attraction representation and user preference profile modeling. Second, we rank attractions based on a linear interpolation among cosine similarities between attraction representation feature groups and user preference feature groups.

[9] https://www.yelp.com/developers/documentation/v2/overview

4.1 Representation of Attractions and User Profiles

According to the gathered useful information, we apply standard IR parsing techniques including turning all words into lower cases, stemming and removing stop words on five parts of attractions' text information respectively, including the title text provided by TREC (denoted as $oritit le$), the title text we crawled (denoted as $title$), the content summary text (denoted as $content$), the positive review text (denoted as $positive$) and the negative review text (denoted as $negative$). Once texts have been parsed, we map terms into ID.

As for attractions, we represent their texts in two ways: Vector Space Model (VSM) and Latent Semantic Indexing (LSI)[10]. First, texts of candidate attractions are represented in VSM under the tf-idf weighting scheme[10]. For LSI, we perform rank-reduced singular value decomposition on the term-document matrix based on VSM representation of each text. Here, we reduce the rank by keeping the largest 20 diagonal entries to represent each text information as a 20 dimension vector. Now, we get 10 semantic feature groups to represent an attraction. We denote feature groups as the name of the text part followed by its representation method. For example, $title\_lsi$ means the feature group that represents the title text we crawled by LSI. Thus, we obtain the feature group set $\mathbf{A} = \{oritit le\_lsi, oritit le\_tfidf, title\_lsi, title\_tfidf, content\_lsi, content\_tfidf, positive\_lsi, positive\_tfidf, negative\_lsi, negative\_tfidf\}$.

As for users, we use features of attractions that a user has rated to represent preference of the user. The user preference representation also shares the same item as attractions, including VSM and LSI representation of 5 text parts. As mentioned before, the ratings of 3 and up show users' interest to attractions. We collect titles provided by TREC ($oritit le$), titles we crawled ($title$), content summaries of the attractions ($content$) of which ratings are equal to or greater than 3 to represent user preference. Reviews from the open web show more details about attractions. Thus, we take reviews as well as other parts to model user preference. For a user, her/ his "positive reviews ($positive$)" collect all of the positive reviews about the attractions that she/he likes, and vice versa, i.e., "negative reviews ($negative$)" collect all of the negative reviews about the attractions that she/he dislikes. Then we represent these text information in ways of VSM and LSI. From an original user profile, we can find two more kinds of information. One is from tags that indicate why the user likes the particular attraction along with attractions added by the user, the other is from several pieces of optional context about the trip. We collect the tags and manually label key words highly related to context of the trip. On average, we pick up 100 key words about each piece of the trip. For example, we label the trip type of "Business" with words "flight", "conversation", "event", "meeting" and etc, and label the season of "Summer" with words "hot", "barbecue", "lemonade", "picnic outside" and etc. We combine the tags from the user and key words related to provided context as an additional text feature to represent user preference. Further, we apply SVM and LSI techniques on this new text to obtain two extra feature groups, denoted as $keyword\_tfidf$ and $keyword\_lsi$. Here, we get 12 feature groups to represent a user's preference.

[10] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

4.2 Ranking Strategy

To generate the final ranking list of attractions, we compute the similarity between attractions and a user preference profile. Representation of user preference profiles and attractions share the same kinds of 10 feature groups. We calculate the cosine similarities between these feature groups. Suppose that the user preference feature vector $R(u)$ and the attraction feature vector $R(t)$ denote any one of VSM representation or LSI representation on the title text provided by TREC, the title text we crawled, the content summary text, the positive review text or the negative review text respectively, the similarity between the two is:

$$sim_f(u,t) = cos(R^f(u), R^f(t)) = \frac{R^f(u) \cdot R^f(t)}{\|R^f(u)\| \|R^f(t)\|}, \tag{1}$$

where f comes from the feature set of $\mathbf{A}$. For feature groups of tags and key words (*keyword*), we calculate the cosine similarity between *keyword* and attraction's positive reviews $positive$, because reviews always show more description about attractions. These two parts are denoted as $keywordp\_lsi$ and $keywordp\_tfidf$.

The final score is based on these similarity scores using linear interpolation. We use the following function to estimate the similarity between a user and a candidate attraction:

$$score(u,t) = w_1 \cdot sim_{keywordp\_lsi}(u,t) + w_2 \cdot sim_{keywordp\_tfidf}(u,t) + \sum_{f \in \mathbf{A}} w_f \cdot sim_f(u,t), \tag{2}$$

where the value of $\mathbf{w} = \{w_1, w_2\} \cup \{w_f\}_{f \in A}$ must be summed up to 1. We use the $score(u,t)$ to rank the attractions on our server. We have submitted two runs during Live Experiment. In the run of IRKM1, we do not consider two *keyword* feature groups, and the weight $\mathbf{w}$ is $w_1 = w_2 = 0$ and $\mathbf{w}_f = \{0.05, 0.05, 0.1, 0.1, 0.2, 0.2, 0.35, 0.35, -0.2, -0.2\}$ (in the same order as $\mathbf{A}$). In the run of IRKM2, the weight $\mathbf{w}$ is $w_1 = w_2 = 0.1667$ and $\mathbf{w}_f = \{0.0333, 0.0333, 0.0667, 0.0667, 0.1333, 0.1333, 0.2333, 0.2333, -0.1333, -0.1333\}$.

5 Batch Experiment

In this section, we describe our approach for generating a personalized rank list of suggested attractions for each "user, context" pair in Batch Experiment. The set of all suggestion requests is released, which provides us a chance to learn like/dislike patterns from user's contextual preference to attractions. First, we derive features about each "user, attraction, context" triplet. Second, we utilize the power of machine learning methods to predict ratings.

5.1 Feature Generation

Given attraction representation and user profile modeling in Section 4.1, we introduce more features to feed our predict algorithm for Batch Experiment. We use all of 12 cosine similarity scores used in IRKM2 as features. Besides VSM and LSI representation, we represent each text part of attraction descriptions with a topic vector

based on Latent Dirichlet Allocation (LDA)[11]. For example, a content summary is mapped into a vector of 20 topics. Then the same operation is conducted on user preference profiles. To get more clues for the triplets, we compute the cosine similarity between user's *keyword* and attraction's negative reviews *negative* (denoted as *keywordn*), and the cosine similarity between user's *keyword* and attraction's content summary *content* (denoted as *keywordc*). So the features derived from text are (24 in total): $\{oritit le\_lsi, oritit le\_lda, oritit le\_tfidf, title\_lsi, title\_lda, title\_tfidf, content\_lsi, content\_lda, content\_tfidf, positive\_lsi, positive\_lda, positive\_tfidf, negative\_lsi, negative\_lda, negative\_tfidf, keywordp\_lsi, keywordp\_lda, keywordp\_tfidf, keywordn\_lsi, keywordn\_lda, keywordn\_tfidf, keywordc\_lsi, keywordc\_lda, keywordc\_tfidf\}$.

In this part, we also want to investigate user preference to URL domains. Thus we extend our features with domain preference under user's attributes such as gender and age. First, we map all distinct domains of attractions into IDs. Second, we count the rating frequency of each domain under each gender value (female and male) and each age group (0-10, 10-20, 20-30, 30-40, 40-50) appeared in the set of all user profiles. Third, we count the frequency of each domain of which the rating is equal to or greater than 3 under each gender value and each age scale. Fourth, we get an average rating for each domain under each gender value and each age scale. For each user, we count the rating frequency, the frequency of ratings equal to or greater than 3 and average rating for each domain. In a "user, attraction, context" triplet, we add these domain-related features according to the domain of the attraction and attribute values (gender, age and specific ID) of the user. Additionally, we add age and gender of the user as features.

5.2 Ranking Strategy

We formulate the problem as a classification problem. Afterwards, we rank the attractions according to their predicted ratings. For the ones with the same ratings, the one with a higher prediction probability provided by classification methods will be ranked higher. We can find a batch of training examples from user profiles. We build features for each "user, attraction, context" triplet. With the features, we try many classical classification methods using the data mining tool Weka[8] and pick up the two with the best performance on training data to submit as our runs. RUN1 uses Ensemble of Nested Dichotomies (END) model[12] that is a meta classifier for handling multi-class datasets with 2-class classifiers by building an ensemble of nested dichotomies, while RUN2 takes Sequential Minimal Optimization (SMO)[13] for training a support vector classifier.

6 Runs and Results

We submitted two runs for Live Experiment: IRKM1 and IRKM2. They both use multiple fields to learn relevance between attractions and users with particular context. Additionally, IRKM2 incorporates user tags and labeled keywords as more feature groups. For Batch Experiment, we submitted two runs: RUN1 and RUN2. The two runs model the contextual suggestion task as a score prediction problem, and explore the contributions of domain preferences and user attributes for the contextual suggestion task, and

apply classification techniques to predict the score of "user, attraction, context" triplets. The difference is that RUN1 uses END model, while RUN2 uses SMO model. For all runs, features benefit from titles, summaries of web page contents, reviews and ratings of attractions we crawled from the open web.

Table 1 shows the overall mean performances of our runs in terms of evaluation measures. First, we observe that our submitted runs in Live Experiment achieve above median performances for both P@5 and MRR. IRKM1 and IRKM2 have P@5 value of 0.3953 and 0.4079, respectively. The MRR of IRKM1 and IRKM2 are 22% higher than average Track Median at 0.4268. IRKM2 is slightly better than IRKM1. It shows that tags and our labeled contextual related keywords help to get the clues of user interests and contextual information. Second, results of RUN1 and RUN2 in Batch Experiment are better than both of IRKM1 and IRKM2. We can refer that machine learning methods and domain knowledges are useful for the task. Third, we see that performances of RUN1 and RUN2 are similar. We learn that feature engineering is vital for classical classification methods.

Table 1. Results of our runs in the contextual Suggestion

	Live Experiment			Batch Experiment		
Name of Run	Track Median	IRKM1	IRKM2	Track Median	RUN1	RUN2
MRR	0.4268	0.5213	0.5461	0.6716	0.6594	0.6535
P@5	0.3163	0.3953	0.4079	0.5090	0.5156	0.4616

7 Conclusions

In TREC 2015 Contextual Suggestion Track, we submitted two runs for both Live Experiment and Batch Experiment. Attraction description and user profiles modeling in all runs take advantage of information from commercial attraction review websites. The information includes: web page content, titles, ratings and reviews of attractions. For Live Experiment, we represented attractions and users using multi-filed texts in the ways of VSM and LSI. In IRKM1, we focus on the similarity between attractions and user historical preference, while we combine user tags and contextual related keywords to obtain the results of IRKM2. For Batch Experiment, we get more features by exploring domain preference under user's attributes such as gender and age, and using LDA to get topic distribution of each type of texts. Then we model the ranking problem as a classification problem using machine learning techniques to learn the ranking list. In the future, we plan to extend our method by considering the category of attractions, and try to aggregate user's category preference for a better user modeling.

Acknowledgments. Part of this research is supported by the Natural Science Foundation of Tianjin (No. 14JCQNJC00600), the Science and Technology Planning Project of Tianjin (No. 13ZCZDGX01098), the National Natural Science Foundation of China

(No. 61300166) and the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20130031120042).

References

1. Li, H., Yang, Z., Lai, Y., Duan, L., Fan, K.: Bjut at trec 2014 contextual suggestion track: Hybrid recommendation based on open-web information. In: The 23rd Text Retrieval Conference (TREC). (2014)
2. Yang, P., Fang, H.: Exploration of opinion-aware approach to contextual suggestion. In: The 23rd Text Retrieval Conference (TREC). (2014)
3. Bah, A., Sabhnani, K., Zengin, M., Carterette, B.: University of delaware at trec 2014. In: The 23rd Text Retrieval Conference (TREC). (2014)
4. Xu, D., Callan, J.: Modelling psychological needs for user-dependent contextual suggestion. In: The 23rd Text Retrieval Conference (TREC), NIST (2014)
5. Kiseleva, J., García, A.M., Luo, Y., Kamps, J., Pechenizkiy, M., De Bra, P.: Applying learning to rank techniques to contextual suggestions. In: The 23rd Text Retrieval Conference (TREC). (2014)
6. Tan, L., Dean-Hall, A., Addala, P., Clarke, C.L.: University of waterloo at trec 2014 contextual suggestion: Experiments with suggestion clustering. In: The 23rd Text Retrieval Conference (TREC), NIST (2014)
7. Samar, T., Bellogın, A., de Vries, A.P.: Better contextual suggestions in clueweb12 using domain knowledge inferred from the open web. In: The 23rd Text Retrieval Conference (TREC), NIST (2014)
8. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD explorations newsletter **11**(1) (2009) 10–18
9. Mihalcea, R., Tarau, P.: Textrank: Bringing order into texts. In: Proceedings of the Association for Computational Linguistics(ACL 2004), Association for Computational Linguistics (2004)
10. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. JAsIs **41**(6) (1990) 391–407
11. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. the Journal of machine Learning research **3** (2003) 993–1022
12. Frank, E., Kramer, S.: Ensembles of nested dichotomies for multi-class problems. In: Proceedings of the twenty-first international conference on Machine learning, ACM (2004) 39
13. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In Schoelkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning. MIT Press (1998)