Word embeddings and Global Preference for Contextual Suggestion

Jian Mo Luc Lamontagne Richard Khoury
Department of Computer Science and Software Engineering, Laval University

Abstract — **In this paper we describe our effort on 2016 Contextual Suggestion Track. We present a new ranking model that captures both global trend of interests as well as contextual individual preference. We trained a regressor on common users data thus it can prioritize popular places and categories. In order to model individual user preference, we introduce word embeddings to represent both user profiles and candidate places as vectors in a same Euclidean space.**

Keywords—Word embeddings, Ranking, Document Similarity, Recommendation System, Point-wise re-ranking

I. INTRODUCTION

In the Contextual Suggestion Track of the TREC conference, participants were asked to develop a system that is able to make suggestions for a particular person with a particular context. The recommendations are contextual as they are based on user's location, profile, and preferences [1].

Two separate tasks are also available in this year conference: live experiment and batch experiment. For the live experiment this year, each group is provided only the user profiles and asked to submit proper recommendations. For batch experiments, several candidate suggestions are provided for participants to re-rank the results and a final precision at 5 is calculated. Unlike last year, teams are no longer asked for setting up a real time server. Thus the focus is on how to generate good ranking for large numbers of candidates. In this paper we present our research group's new approach to the ranking problem demanded in the contextual suggestion task this year.

We participated in both tasks of this track. For the live experiment, we use the same framework as last year which was composed of an ElasticSearch engine for the initial selection and a customized ranking module to re-rank the results [2].

For the batch experiment this year, we adopted a new ranking model combining a word2vec based ranker and a global preference regressor.

We use Precision at 5 to present the results obtained for the evaluation of our system configurations. Our P@5 results of Phase 2 reached 50.7% comparing to TREC median 39.3% which demonstrates the effectiveness of our approach.

II. OUR APPROACH

A. System Framework

To perform our experiments, we relied on the following components:

- i Information gathering:
  - a) Crawlers
- ii Recommendation System
  - a) User profile modeling
  - b) ElasticSearch and Customized Queries
  - c) Ranking Model
    1. Word embeddings Model
    2. Global Preference Regressor

The entire framework is illustrated in Figure 1 and each component is described in detail in the following sections.

B. System Overview

Figure 1 showed the overview of our system. We store both our corpus and TREC corpus in ElasticSearch. In the live run we query ElasticSearch for initial candidate suggestions. For details of this part, please refer to our last year paper [2]. Then a pre-trained regressor will score each candidate suggestion according to their popularity among all people. Word vectors are generated from both user profile and candidates then candidates will be scored by their distance to the profile vector. Linear interpolation is used to combine the two scores.

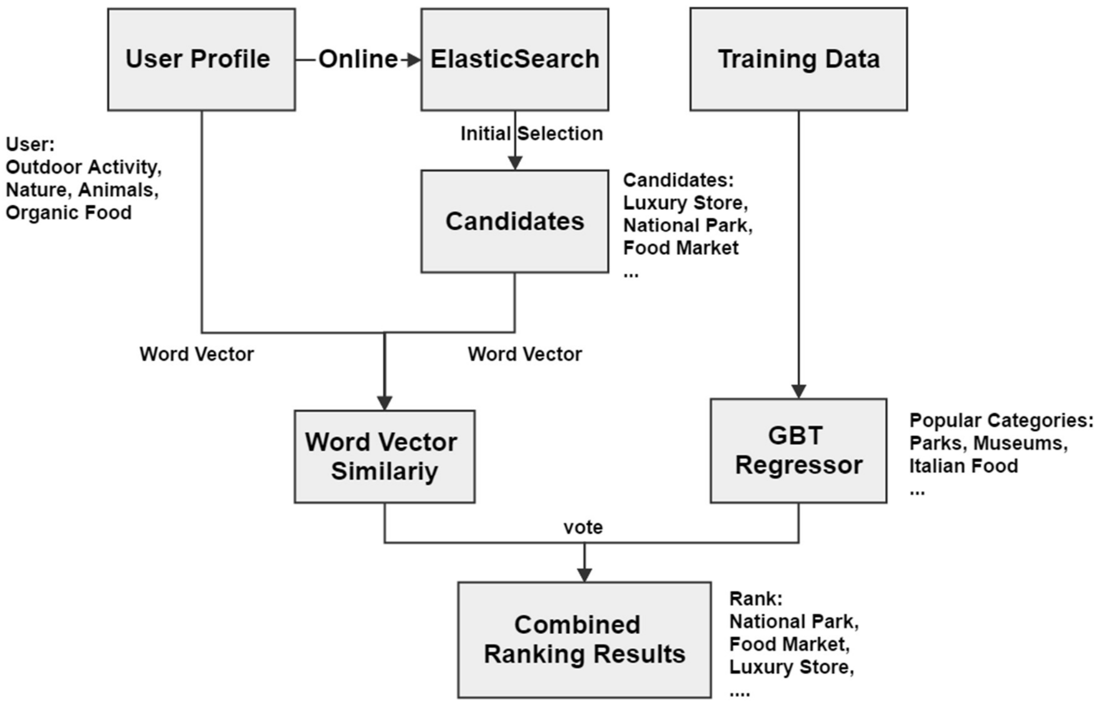

Fig 1. Overall framework for contextual recommendation

III. Information Gathering

We use our old data crawled last year since the application domain and context this year remains the same. Most of our data are coming from Yelp and TripAdvisor [2].

The crawler extracts structural information such as *rating, open hours, price, review count, category for each candidate attraction from the webpages*. Business information such as *has Wi-Fi, has parking lot, smoking allowed* are also extracted and stored into the database.

User reviews were often too numerous to download entirely. Hence we just get the most popular reviews of each attraction, i.e. the first page of reviews. Positive and negative reviews are downloaded separately and stored in two different database tables.

We also used the crawled corpus provided by TREC this year. However these crawled files are raw html and have a lot of noise such as html tags and JavaScripts. So we extract only text and some metadata as representation for each site.

IV. Recommendation System

A. User Profile Modeling

We create the user's positive profile by merging the positive information from all the examples our user likes, and likewise build the negative profile by merging the negative information from the examples our user dislikes. The intuition is that the preferences of one user are reflected by the attractions he/she likes and dislikes. Consequently, we can compare the user profiles with every candidate suggestion in the database and rank them by similarity. For instance, if one candidate suggestion has many elements in common with the positive user profile, this candidate obtains a higher ranking score. In contrast, if the attraction is very similar to the negative user same profile, its ranking score is penalized and will be very low.

According to the task defined for this track, one user might provide 6 different rating:

4: Strongly interested
3: Interested
2: Neutral
1: Disinterested
0: Strongly disinterested
-1: Website didn't load or no rating given

We selected rating 4 and 3 as positive and 1 and 0 as negative. Ratings 2 and -1 were taken as neutral and simply ignored.

Formally, the user profile can be expressed as:

$$Profile_{Pos} = \bigcup REP(Pes_{ik})$$

$$Profile_{Neg} = \bigcup REP(Nes_{ik})$$

Where Pes_i is positive example suggestion i, and Pes_{ik} is element k (categories, tags, positive reviews, business info) of this positive suggestion.

$REP(Pes_{ik})$ defines a special representation or form of the element.

B. *Initial selection by Elasticsearch*

Once the user profile is built, we formulate a customized query to search our ElasticSearch database. For example, let's suppose the user likes Mexican food, dislikes Japanese food, appreciates Wi-Fi and parking lot and hates smoking. We use a bool boosting query to wrap up the elements of the user profile into one query, which can then be sent to ElasticSearch to retrieve relevant new attractions.

These returned 100 attractions are used as our initial selection pool for later ranking. For further details on this part, please refer to our last year paper [2].

C. *Ranking Model*

Initially selected candidates are then re-ranked by the ranking model. There are two major module in our ranking model — Word embeddings Model and Global Preference Regressor. The first one is used to capture individual interest.

1) *Word embeddings Model*

Word embeddings are known for performing well at analogical reasoning. Example given by Mikolov [3] — king-man=queen-woman — showed that word embeddings could reason analogy or indirect relationship between words.

This property of word embeddings makes us wonder if it could be applied to recommendation or ranking problem. In Contextual Suggestion, a user who likes organic food and yoga would probably enjoy vegetarian restaurant and modern art while may dislikes fast food. Word embeddings model seems promising in finding subtle relationships in these occasions.

So we introduce word embeddings as special representations for candidates aiming to find semantic or contextual similarity between users and attractions.

Firstly, for every attraction we extract a list of nouns as representation of the place. For example, a Macdonald can be represented by words list — Fast-food, American, Restaurant. And a user can be represented by positive keywords and negative keywords as stated in user profile modeling section.

Then, we use Genism [4] trained on the GoogleNews corpus [5] to generate word vectors for each keywords in list.

A rather intuitive method is adopted to merge word vectors by summing all up into one. So in the end every attraction is represented by one vector while user profiles are matrices composed of rows of attraction vectors multiplied by their rating.

Finally we calculate similarity between attraction vectors and user matrix. The first method is to sum up user matrix vertically to merge it to a single vector and then calculate the dot product of attraction vector and user vector.

The second method is only calculate the one vector in user matrix which is nearest (smallest inner product) with attraction vector. The algorithm is simply multiplying user matrix and attraction vector then taking the smallest number in the vector.

Both of these two methods achieved similar results in our test runs. We used the first method in our submitted runs.

2) *Regressor Trained on Global Preference*

After examining the user data and candidate attractions, we found that sometimes categories appearing in candidates does not show in user profiles. For instance, one user only rated restaurants but his preference towards sports and activates remains unknown. The situation is common in recommendation problems.

Our approach to this missing profile problem is to recommend popular places all people favor. A Gradient Boosting Tree is trained on 2015 TREC data with features of *ratings, review count, category* and *relevant index* in the list. The relevant index means the order of appearance of the candidate place. We introduce this feature to model declining interest of user namely the law of diminishing marginal utility.

The trained regressor would prioritize the popular category and attractions and those who appear earlier in the candidate list.

3) *Combine the two models*

An overall ranking is given by arbitrarily averaging both ranking from the two modules. That is to say both modules have the same proportional votes for the final ranking. The weighting parameters might be learnt by another training process which we didn't apply in these experiments.

V. Results and Analysis

A. *Evaluation Metric*

An attraction is considered relevant for P@5 if it has a geographical relevance of 1 or 2 and if the user reported that both the description and document were found to be interesting (3) or strongly interesting (4). A P@5 score for a particular topic (a profile-context pair) is determined by how many of the top 5 ranked attractions are relevant, divided by 5. [1]

B. *Submitted Runs*

Three runs were submitted to the competition:

- LavalIVA_live1, live run
- LavalIVA_batch1 is batch run 1, which applies only the global regressor.
- LavalIVA_batch2 is batch run 2, using only the word vector model.
- LavalIVA_batch3 is batch run 2, which combined the two modules.

Table 1: TREC CS 2016 results

Run	P@5
Live_1	0.27
Batch_1	0.4345
Batch_2	0.4276
Batch_3	0.5069
TREC median	0.393103

Table 1 shows our final results. The low score of live run was caused by a sorting problem in our code which was fixed later in batch runs. So we will mainly discuss about the three batch runs.

As can be seen from the table, all the three batch runs scored higher than TREC median.

Batch_1 used only the Global Preference regressor which only recommend popular places for everyone regardless of their personal interest. Surprisingly this model scored above average even though it does not consider user profile at all.

Batch_2 used only the word embeddings module which only focus on the individual user preference.

Batch_3 is a result of the combination of the two models. The fact this run scored highest indicates that the two modules cover different aspect of the final recommendation and gives a better result we.

Here we study some cases to show how these two modules worked respectively showed in Table 2.

Table 2. Model on 2015 validation set

User/Runs	median	b1	b2	b3
740	0	0.6	0	0
774	0	0	0.4	0
705	0.4	0.4	0.2	0.8

For user 740, only global regressor could make correct recommendations partly because the user was not consistent with his profile and favors popular places when re-rated.

User 774 is picky and only has two very positive scores on beer bars so the word embeddings module correctly selected 5 beer bars for these user while global regressor fail to guess it right.

Both modules get parts of the user 705 preferences so the combined model gave highest accuracy. For details on how each run scored please check the table in appendix at the end of the paper.

C. *Results on Test set*

We also tested our model on 2015 data. A noticeable result showed in Table 2 is that regressor which models the declining user interest scores much better than the one without. So we suggest user do have a tendency to like top results more than those placed at very bottom of the list.

A decay model is used in our submitted runs. However we did not observe such behavior in 2016 data after the validation results released.

It was later revealed that the 2016 request list had been random shuffled in case this vulnerability being exploited again as some team just returned the original requests and still got good scores at 2015 TREC conference.

Table 3. Model on 2015 validation set

Run	P@5
NoDecay	0.581
DecayInterest	0.6932

VI. Conclusion and Future Work

In this paper, we have presented our new approach to contextual suggestion. Both the global regressor and word embeddings model showed effectiveness respectively and so the combined model could capture individual preference when training data is sufficient while provide popular recommendations for user with less data.

Specially, we showed that contextual recommendation problem can be tackled by cultivating analogical power of word embeddings. Even naïve approach at this stage showed good results.

We only sum up word vectors to represent the paragraph vector in which the order of words is lost. A more fine grained representations could be obtained as Doc embeddings or paragraph vector proposed in [6].

While word embeddings works well on textual data such as comments and categories, a different approach would be to use word2vec technique to encode venues just by their interrelationship [7].

RNN or other models that also vectorize documents fit well to our model and remains to be tested in future experiments.

REFERENCE

[1] A. Dean-Hall, C. Clarke, J. Kamps, P. Thomas and E. Voorhees. Overview of the TREC 2015 Contextual Suggestion Track. In Proceedings of TREC'15, 2015.

[2] Jian Mo, Luc Lamontagne, Richard Khoury Laval University and Lakehead University Experiments at TREC 2015 Contextual Suggestion Track. In Proceedings of TREC'15, 2015.

[3] T. Mikolov, W.-T. Yih, and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations," pp. 746–751, 2013.

[4] https://radimrehurek.com/gensim/

[5] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

[6] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," Int. Conf. Mach. Learn. - ICML 2014, vol. 32, pp. 1188–1196, 2014..

[7] M. G. Ozsoy, "From Word Embeddings to Item Recommendation," arXiv, 2016.

VII. APPENDIX

user_id	median	batch1	batch2	batch3
700	0.6	0.4	0.6	0.4
701	0.4	0.6	0.4	0.4
703	0.4	0.2	0.4	0.8
704	0.4	0.6	0.4	0.8
705	0.4	0.4	0.2	0.8
706	0	0.2	0	0
712	0	0	0.2	0
715	0.6	0.4	0.6	0.6
718	0.4	0.8	0.2	0.6
719	0.4	0.6	0.4	0.6
720	0.4	1	0.4	0.8
721	0.4	0.6	0.6	0.8
722	0.4	0	0.6	0.6
723	0.4	0.4	0.4	0.6
726	0.6	0.4	0.4	0.8
727	0.6	0.4	0.6	0.6
728	0.6	0	0.8	0.4
729	0.4	0.6	0.6	0.6
731	0.4	0.8	0.4	0.8
732	0.4	0.6	0	0.8
733	0.6	0.6	0.6	0.4
734	0.8	0.6	0.8	0.8
740	0	0.6	0	0
743	0.6	0.4	0.6	0.8
744	0.6	0.6	0.4	0.8
745	0.6	0.6	0.6	0.6
746	0.6	0.2	0.8	0.8
753	0.2	0	0.2	0.2
754	0.4	0.6	0.6	0.6
756	0.8	1	0.6	0.8
759	0.2	0	0.6	0.6
760	0.2	0	0	0
761	0.6	0.6	0.4	0.6
762	0.4	0.4	0.4	0.6
763	0.6	0.6	1	0.6
764	0.6	0.4	0.6	0.6
765	0.2	0.4	0.2	0
768	0.6	0.6	0.8	0.6
773	0.2	0.2	0.2	0.4
774	0	0	0.4	0
776	0.4	0.6	0.4	0.6
777	0.4	0.6	0.6	0.6
779	0.4	0.4	0	0.4
781	0.2	0.4	0.4	0.6
788	0.2	0.2	0.2	0
791	0.2	0.2	0.6	0.2
792	0.4	0.8	0.2	0.2
794	0.2	0.8	0.4	0.8
795	0.4	0.8	0.6	0.6
796	0.2	0	0.2	0.2
797	0.4	0.6	0.2	0.6
798	0.2	0.4	0.2	0.8
804	0.2	0	0.4	0.2
810	0.2	0	0.4	0.6
811	0.2	0.2	0.4	0
813	0.2	0.4	0.2	0.2
814	1	1	1	1
902	0.4	0.4	0.4	0.2