Employing open-web for Contextual Suggestion using tag-tag similarity

Sainyam Kapoor, Manajit Chakraborty and C Ravindranath Chowdary

Department of Computer Science and Engineering,
Indian Institute of Technology (BHU), Varanasi-221005, India
sainyam.kapoor.cse12@iitbhu.ac.in
cmanajit.rs.cse14@iitbhu.ac.in
rchowdary.cse@iitbhu.ac.in

Abstract. The TREC 2016 Contextual Suggestion task aims at providing recommendations on points of attraction for different kind of users and a varying context. Our group DPLAB_IITBHU tries to recommend relevant point-of-interests to a user based on the information provided on the candidate attractions and her past preferences. We employ open-web information in a novel way to capture the best possible setting for a user's tastes and distastes in terms of tag scores. The scores are then ranked using a heuristic to suggest the most pertinent attraction to the user. One of our methods exceed the TREC-CS 2016 median of the standard evaluation scores of all participant runs, which reflects the effectiveness of our approach.

Keywords: Contextual Suggestion, Points-of-interest, Wikipedia

1 Introduction

The TREC Contextual Suggestion track investigates search techniques for complex information needs that are highly dependent on context and user interests. The task is to provide suggestions to users based on their personal interests as well as their contexts. In our last year's attempt we tried to capture user interests in terms of tags and check its similarity to the profile of the users. This technique produced decent results but we did not consider the contexts of user's visits. Our aim this year was to specifically address that issue and in the process increasing the efficacy of the recommender system. Our proposed method ranks candidate suggestions based on the similarity of the tags between user visited attractions and the candidate suggestions. The candidate suggestion tags in turn are ranked based on their admittance to contexts. We prepare three different heuristics to capture the best possible setting so as to model the user's taste and corresponding catering to those tastes.

We reformulated the problem of contextual suggestion as the following question: "*Can we find all the correct tags for an attraction for a user based on the context she's provided?*". To answer the question we took the help of open-web *i.e.* Wikipedia. We scored the mutual similarity of all the tags provided in the tagset and the contexts against themselves. The tags were then ranked based on their similarity scores and then a further matching of the candidate suggestion tags against the user profile attraction were carried out. The final score was computed using three different heuristics as explained in the next section, which yielded three runs submitted this year.

The details of our recommendation model include data pre-processing, tagging, scoring and finally ranking the candidate suggestions.

2 Model

In this section we describe the methodology used to generate the suggestion list for users based on their profiles and the given context. While the data preprocessing and tagging methods are same across the three runs, the difference lies in the heuristic used to calculate the ranks of the candidate suggestions. The detailed work-flow is as follows:

- **Data Preprocessing**:
  (i) For each of the 224 tags in the provided tag list, we extract its corresponding Wikipedia page. We perform the same task for each of the 15 context tags (*i.e.* Trip type, Season *etc.*).
  (ii) We construct a matrix $\mathcal{M}_{TT}$ where both the rows and columns consist of Wikipedia pages corresponding to the tags. Each of the elements in the matrix represent a similarity score between the wiki pages of the tags, in effect implying the measure of closeness of a tag i to tag j $(i \neq j)$. We have considered only the visible text from the wiki page excluding the title. For computing similarity, cosine similarity was used where *idf*[1] was computed on the wiki pages (documents) for tags. Additional cleaning of the text from the wikipedia pages was performed to remove copyright and other irrelevant text.
  (iii) Next, we pre-process the provided descriptions of the user profile and candidate suggestion attractions by removing sentences which contains irrelevant and garbled words.
  (iv) The stop words and special characters (!@#$%^etc.) except punctuations were removed and only intelligible words were retained in both the tag documents (wiki pages) and attraction descriptions. Also, stemming and lemmatization was carried out.

- **Modeling User Profile**:
  The first step towards an efficient recommendation system is to understand the user's needs and tastes. Incorporation of this knowledge can positively affect the quality of recommender system. Since, we don't have any personal information about the users, we have to rely on the rating and tags of the attractions the user has previously visited.

[1] The Stanford NLTK [1] was used for this purpose.

(i) For each attraction in the user profile, we tag the attraction with n tags. The tags are associated based on its similarity to the description of the attraction. To determine the similarity between the tag to be assigned and the attraction, we construct a matrix $\mathcal{M}_{TD}$, whose elements represent the similarity between the Wikipedia page of the tag under consideration and the attraction description. For example, let us say, attractions are denoted by $x_1, x_2, x_3, ..., x_m$ and the tags from the predefined taglist are denoted by $t_1, t_2, t_3, ..., t_{224}$. Then the matrix $\mathcal{M}_{TD}$ would look something like:

Once again, cosine similarity was used for computation. By attraction,

Table 1. Attraction to Tag similarity matrix $\mathcal{M}_{TD}$

	Tags			
Attraction	t_1	t_2	...	t_{224}
x_1	0.045	0.003	...	0.135
x_2	0.246	0.024	...	0.008
:	:	:	:	:
x_m	0.251	0.306	...	0.019

we mean the description associated with the attraction. The figures presented in Table 1 are indicative and not the original figures.

(ii) Next, we sort the tags based on their similarity scores for each attraction in the user profile. We retain only top k tags. Here, $k = 10$. A similar procedure is carried out for each of the candidate suggestions as well.

(iii) Now, we take a weighted count of each unique tag across the set of attractions for a user profile. The weights differ from approach (run) to approach as explained later.

(iv) Lastly, we retain the top-20 tags based on the value of its weights. These tags are henceforth referred to as *user-profile tags*. These user-profile tags capture the essential needs of the user. In other words, these selected tags model a user specific taste. Further, refinement of the user-profile tags is carried out by re-ranking the tags based on the contexts. This involves ranking the 20 tags based on their $\mathcal{M}_{TT}$ score. In case of missing context, the original order of the tags is retained. The reason behind this exercise is to bring out the order among the tags which the user may prefer according to the specified context.

– **Ranking candidate suggestions**:
The final step involves ranking of candidate suggestions based on its score. The score is calculated on the basis of proximity of the candidate suggestion tags to the user-profile tags. To determine this proximity we first construct a matrix $\mathcal{M}_{TC}$, where each element represent the similarity of the candidate suggestion tags to the user-profile tags. We achieve this by assigning each tag present in the candidate the reciprocal rank of the corresponding user-profile tag list. In case, the candidate tags did not appear in the user-profile

tag list, we assign it a score of zero. Finally, we take the summation of all the tag scores for each candidate suggestion and term it as the *candidate score*. The suggestions are then ranked based on the candidate scores. This method is common to all the three approaches. The difference lies in the tagging mechanism and assignment of weights to user-profile tags which we discuss next.
Approach 1 (RUN 1): In this approach we assume that the user has not provided any tags. Thus, we tag all the attractions in user profile and candidates uniformly without bothering for the user-defined tags. This was done to ensure that there are no discrepancies involved in mixing the tags provided by users and the ones assigned by us. In other words, we eliminated the dilemma of choosing the best tags among the two sets, in effect reducing any error that may creep in due to misjudging the appropriate tag for an attraction.
For determining weights of tags we take the help of ratings that the user has assigned to each attraction. We segregated the ratings into two levels:

- +1 for all the attractions that have been rated 4 or 3 by the user, suggesting her liking for that particular attraction.
- -1 for all the other attractions that have been rated less than 3 but greater than 0, expressing user's disinterest in the same.

For all other attractions that the user rated -1, we simply discarded them. Each of tags associated with an attraction was initially assigned the same rating *i.e* if the attraction has been rated 4, all the tags would carry an initial weight of +1. Similarly, all the tags associated with a low rated attraction would carry a weight of -1. Now, for each of the unique tags, we summed the weights associated with it. The tags are then ranked based on their final weights w_f.

Approach 2 (RUN 2): This approach is similar to Approach 1 differing only in the fact that we considered both the user-defined tags along with the tags assigned by us. We limited the number of tags for attractions to 10 as stated earlier. Now, say, the user has already provided p tags, then we assign the rest $10 - p$ tags using our method explained above. The consideration of the user tags was done to ensure that we were not completely alienating ourselves from user specified choices, a risk we run if we concentrate only on our tagging method. Rest of the method is in line with Approach 1.

Approach 3 (RUN 3): This approach differs from the other two in the fact that here we consider only the frequency of the tag in the user profile attraction set instead of assigning two-level weights. By doing so, we want to capture the user's tastes without accounting for the rating associated with the tag. We go by the simple notion that *"the more frequent a tag is, the more chances that the user likes that facility"*.

3 Results and Discussion

We employed three techniques for capturing the user tastes and deliver a recommendation list that caters to her needs. All the three techniques employed open-web information in the form of Wikipedia pages associated with the tags and designed heuristics to calculate the score of each candidate suggestion accordingly. From Table 2 it is clear that Approach 1 performed better than the other two approaches. In fact, our first run surpassed the median of this year's evaluation results against all three metrics.
A comparative assessment of our methods against other teams could not be performed due to unavailability of their technical description and evaluation results. But as far as performance is considered, our first approach performs slightly better than the median results as listed in Table 2.

Table 2. Evaluation Results

	NDCG@5	P@5	Reciprocal Rank
Approach 1 (iitbhu01)	0.2757	0.4138	0.6298
Approach 2 (iitbhu04)	0.2325	0.3310	0.5367
Approach 3 (iitbhu05)	0.2106	0.3034	0.4921
Average TREC Median[2]	0.2562	0.3931	0.6015

References

1. Edward Loper and Steven Bird. NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 63–70. Association for Computational Linguistics, 2002.

[2] TREC 2016 Contextual Suggestion Batch Experiment Results