Contextual Suggestion using tag-description similarity

Manajit Chakraborty, Hitesh Agrawal, Himanshu Shekhar and C Ravindranath Chowdary

Indian Institute of Technology(BHU), Varanasi-221005, India
cmanajit.rs.cse14@iitbhu.ac.in
hitesh.agrawal.ece13@iitbhu.ac.in
himanshu.shekhar.ece13@iitbhu.ac.in
rchowdary.cse@iitbhu.ac.in

Abstract. In this paper, we present our approach for the Contextual Suggestion track of 2015 Text REtrieval Conference (TREC). The task aims at providing recommendations on points of attraction for different kind of users and a varying context. Our group DPLAB_IITBHU tries to address the problem from the perspective of how relevant the attractions are based on user profiles and rank them based on two similarity measures– *wup* similarity and another similarity measure proposed by us.

1 Introduction

The 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. To tackle this problem, we propose to rank candidate suggestions based on their similarity to the personal profile to the contexts (*i.e.*, geographic and temporal information *etc.*). The ranking function is computed based on the similarity between a suggestion and the places that the user like and the dis-similarity between the suggestion and the places disliked by the user. The similarities are computed based on the description of the suggestions. The paper describes the first participation of Indian Institute of Technology (BHU) in 2015 TREC Contextual Suggestion track. The descriptions for suggestions were gathered from the *url*s provided in the collection. We got two runs for the final results based on two similarity measures. The details of our recommendation model include data pre-processing, rating 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. It is to be noted that

we used a similar method to generate the recommendation list, the difference being the technique used to calculate similarity between the attractions. The work-flow is as follows:

- **Step 1**: *Extracting description of the attractions provided in the batch experiment candidate suggestions and pre-processing them (30 for each user)*

  (i) First of all, we used a script written to extract information from the *url* of each attraction provided in the candidate suggestion list.
  (ii) For attractions (and its corresponding *url*s) for which the above script couldn't pull the required information due to page redirection or other accessibility issues, we used the Foursquare API[1]. Using this API we extracted the category of the attraction in question along with the user reviews mentioned there. Henceforth, we will refer to the information extracted from *url*s as 'description' unless otherwise mentioned.
  (iii) Next we pre-processed the descriptions by removing sentences which contained irrelevant and garbled words.
  (iv) The stop words and special characters (!@#$%^etc.) were removed and only intelligible words were retained.
  (v) The filtered descriptions were then tagged using a Parts-of-Speech (POS) tagger [1] in order to keep only nouns, verbs, adverbs and adjectives while discarding the rest.

- **Step 2**: *Finding the tags of each candidate suggestion*:
  Our whole procedure depends on matching user provided tags with attraction description. Hence, this step is of prime importance, as we try to capture the essential elements from the descriptions in terms of tags.

  (i) We generated a list *for_similarity_list* which contained tags provided with TREC 2015 Contextual Suggestion track dataset and processed it.
  (ii) From observation we found out that some tags were very specific while others were pretty generic. To distinguish between them, we marked each tag as either 0 or 1. If the tag is marked with 0, it implies that the tag has to be matched directly with the attraction description. On the other hand, if it is marked with 1, then instead of directly using the tags we expand it using *synsets* from Wordnet[2] and then used for matching with the descriptions. This step is done so as to encompass related terms such as 'cuisine' with 'food' etc. For bi-gram tags, each word in the tag was expanded using synset before matching.
  (iii) In cases of suggestions for which tags weren't explicitly provided, we extracted the description from the *url* and matched them across the list of all tags. The most similar ones were used as tags for that particular attraction. The similarity matching was done using wup_similarity [3] with a threshold of 0.95.

[1] https://developer.foursquare.com/

(iv) Finally, all the tags were matched against candidate suggestion descriptions and the matching ones were appended in a list of that particular suggestion.

– **Step 3**: *Assigning rating to each candidate suggestion*:
Approach 1 (RUN 1): The complete process discussed below is carried out for each user profile.
(a) A dictionary $\mathcal{D}_1$ is prepared from all the rated attractions with each entry having the highest rated attraction tags first, followed by next highest rated attraction tags and so on. The outline of such a dictionary is provided in Table 1.

Table 1. Dictionary of Tags with Ratings

Rating	Tags
4	⟨tags from attraction xv⟩ ⟨tags from attraction yv⟩ ...
3	⟨tags from attraction xx⟩ ⟨tags from attraction yx⟩ ...
2	⟨tags from attraction xy⟩ ⟨tags from attraction yy⟩ ...
1	⟨tags from attraction xz⟩ ⟨tags from attraction yz⟩ ...
0	⟨tags from attraction xu⟩ ⟨tags from attraction yu⟩ ...
-1	⟨tags from attraction xw⟩ ⟨tags from attraction yw⟩ ...

(b) Now each candidate suggestion description was matched against this dictionary $\mathcal{D}_1$. This was done to rate the particular attraction with a rating having maximum number of matching tags. So, suppose a suggestion description was matched against the dictionary and the following result came out (Table 2):
Then the candidate suggestion is assigned a rating of 3 since there are

Table 2. Result of matching a candidate suggestion against the dictionary

Rating	Number of matched tags in a list
4	5
3	6
2	6
1	2
0	3
-1	1

6 matched tags. In case of a tie, the highest rating among the choices is assigned.
(c) Similarly, for the rest of the 29 candidate suggestions a list of (Rating, No. of Matched tags) is prepared. The list was sorted first on the basis

of rating and then further within each rank on the basis of number of tags matched.

(d) If the matching of a candidate suggestion's description with tags returned 0, then it wasn't included in the list. This is the prime reason why in the evaluation our approach failed to retrieve any suggestion for few profiles.

Approach 2 (RUN 2): Similar to Approach 1, this process is carried out for each profile as well.

(a) A dictionary was prepared containing a list for each rating. Each element in this list represent the count of tags with rating X where X varies from 4 to -1. *E.g.*, for a rated attraction if the rating was 3, then in list corresponding to rating 3, the count of tags was incremented by 1.

(b) Once all the rated attractions were processed, the list corresponding to each rating was normalized , *i.e.*, count of each tag is divided by total number of count in that rating. Now, sum of count of all tags in a particular rating equals to 1.

(c) Now each candidate suggestion tags were matched to each rating list and the corresponding score of each tag was added to give the total score of that suggestion. *E.g* if tags a, b, d occurs in a candidate suggestion, for each rating, their corresponding scores were added and the rating with maximum score was assigned to that suggestion.

(d) If no matching tags were found for a particular suggestion, it wasn't added to our list. In case of collision the highest rating was assigned.

(e) The list was sorted first on the basis of rating and then within each rank on the basis of total score obtained.

3 Discussion

We employed two techniques for matching tags with candidate suggestion descriptions to find the best possible suggestion for a particular context. These suggestions were then ranked according to the user preferences as provided in the profiles. Among the two approaches, Approach 2 performed better than Approach 1. This can be attributed to the fact that the ranking system in Approach was more fine-grained with each tag carrying its own weight unlike in Approach 1 where either the tags were considered or rejected. So, the number of retrieved suggestions were generally greater in Approach 2. Since, we weren't provided with the evaluation results of other teams, we can't perform a comparative assessment. But, speaking in terms of improvement our approaches perform better than the median results as listed in Table 3.

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*

[2] TREC 2015 Contextual Suggestion Batch Experiment Results

Table 3. Evaluation Results

	P@5	R-Precision	Reciprocal Rank	MAP
Approach 1 (IITBHU_1)	0.5308	0.4941	0.6760	0.5581
Approach 2 (IITBHU_2)	0.5365	0.4969	0.7030	0.5639
Average Median[2]	0.5090	–	0.6716	–

natural language processing and computational linguistics-Volume 1, pages 63–70. Association for Computational Linguistics, 2002.

2. George A Miller. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
3. Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.