A Simple Context Dependent Suggestion System

Dwaipayan Roy, Ayan Bandyopadhyay, Mandar Mitra Computer Vision and Pattern Recognition Unit Indian Statistical Institute 203, B.T.Road, Kolkata, India - 700108. {dwaipayan.roy, bandyopadhyay.ayan, mandar.mitra}@gmail.com

Abstract

The Contextual Suggestion Problem focuses on search techniques for complex information needs that are highly dependent on context and user interest. In this paper, we present our approach to providing user and context dependent suggestions. The official evaluation scores for our submitted run are reported and compared to the overall median scores (across all 34 runs).

1 Introduction

This is the second year of the Contextual Suggestion Track at TREC. The goal of the track was to investigate search techniques that consider both user-preference and context. User-preferences are represented via past history of personal preferences. Context is provided as a particular location (latitude, longitude) for which the suggestions need to be made. A system is expected to return a ranked list of at least 50 suggestions for each given user-context pair. In order to solve the problem, we need to address the following issues:

- how to determine the potentially useful categories of suggestions for a user;
- how to fetch candidate suggestions from these categories;
- how to find detailed information about these candidates;
- how to rank the candidates based on preference and context.

2 Our method

The dataset provided consists of two parts:

- 1. **Training data**: 562 profiles containing user preferences for about 50 example suggestions; and
- 2. Test data: 50 geographical contexts for which suggestions are required.

We use P to denote the set of all 562 user profiles, while C denotes the set of all 50 contexts for which suggestions have to be made.

2.1 Understanding user preferences

The preferences of a user can, naturally, not be properly characterised using the limited amount of rating information available per user. We make the simplifying assumption that a low score given by a user to an example suggestion is indicative of the user's general disinterest in that *category* of suggestions. For example, if a user gives a poor score to a particular zoo, we assume that the user is not interested in zoos in general. We used the Google Map API to obtain category information for the suggestions provided as training examples. A total of 18 different categories of suggestions were identified.

For each user-suggestion pair in the training data, two scores were provided: one for the web-site of the example attraction (url-score), and another for the brief description of the attraction (description-score). The scale for these scores was shifted from 0–4 to 1–5. The shifted *url-score* and *description-score* were combined using the harmonic mean. The overall preference of a user for each category was obtained by aggregating her scores over all example suggestions belonging to that category. The details of this process are given in the algorithm below. Any category with a non-zero preference score for a user is regarded as a *permissible* category for that user.

```
Input: a user profile p from P
for each type t from among the 18 categories identified do
    score[t] = 0;
    number_of_examples[t] = 0;
end
for each e from the set of 50 example suggestions rated by p do
    t = type of e as specified by Google Map API;
    score[t] += harmonic-mean (description-score(e), url-score(e));
    number_of_examples[t]++;
end
for each type t from among the 18 categories identified do
    preference[p,t] = score[t] / number_of_examples[t];
end
```

Algorithm 1: User preferences for various categories

2.2 Gathering suggested places

For each user (or profile) p, and each context (or location) c, we used the Google Map API[1] once again to create a list $L_{p,c}$ of potentially interesting places that may be returned to the user. Given the location c of a user p, a set T of permissible types for p, and a distance d, this API returns a ranked list $L_{p,c}$ of at most 60 places from the permitted types within a distance d from c. We start with d = 3 km. If the returned list $L_{p,c}$ contains fewer than 60 results, this limit is increased to 5 km.

2.3 Ranking based on physical distance and preference

For each place in the list $L_{p,c}$ constructed above, its distance from the user's current location and the user's preference for that type of place (as determined in Section 2.1) are combined to obtain a composite score S. The list of places is then ordered on the basis of S, and the top 50 places are selected as suggestions.

Input: a user profile p, and a context cfor each suggestion s in $L_{p,c}$ do t = type of s as given by Google Map API ; $S(s) = 0.5 \times preference[p, t] + 0.5 \times Distance(c, s)$ end

Algorithm 2: Ranking results

2.4 Gathering description of the suggestions

Finally, for each suggestion s returned to the user, we also need to provide a brief description. Given s, its name is submitted to the Google Web Search API[2]. The top-ranked snippet retrieved in response to this query is used as the description.

3 Experimental results

Systems are evaluated using three metrics: P@5, MRR and TBG (Time Based Gain). In this section, we present the evaluation results for our run, and compare it with the median results for each context.

Figures 1, 2 and 3 graphically compare the P@5, MRR and TBG values for our run (isirun) with the corresponding median score for each context. While our run has mostly achieved above median scores for MRR and TBG, the P@5 scores for our run are less satisfactory.

	P@5	MRR	TBG
isirun	0.3650	0.5165	1.6278

Table 1: Overall performance of our submitted run

Table 1 presents the overall performance figures for **isirun** (averaged over all judged profiles and contexts).

4 Conclusion

In this paper, we have explained our approach for the TREC 2013 Contextual Suggestion Track. The results show that our system generally performs better than the median with respect to MRR and TBG. There appears to be significant scope for improvement in the P@5 score, however.

References

- [1] Maps API Google Developers https://developers.google.com/maps/
- [2] Google Web Search API https://developers.google.com/web-search/



Figure 1: P@5 of isirun compared with Median on each context by averaging all profiles for that context



Figure 2: MRR of isirun compared with Median on each context by averaging all profiles for that context



Figure 3: TBG of isirun compared with Median on each context by averaging all profiles for that context