WaterlooClarke: TREC 2015 Contextual Suggestion Track

Hella Hoffmann
University of Waterloo
hrhoffmann@uwaterloo.ca

Pragnya Addala
University of Waterloo
paddala@uwaterloo.ca

Charles L. A. Clarke
University of Waterloo
claclark@uwaterloo.ca

November 2015

1 Introduction

In this work we present a first attempt at developing a live system to solve the problem presented in the TREC 2015 contextual suggestion task[1]. The goal of this task is to tailor point-of-interest suggestions to users according to their preferences [3]. We present how we gathered data for the candidate points-of-interest, filtered some of the candidates and built a live system to return suggestions that would most likely interest the specific user.

As part of TREC 2015, the contextual suggestion track is running for the fourth time [3, 4, 2] and this is the first time that the participants were required to build a live suggestion system. The general idea is to be able to make real-time suggestions for a particular person (based upon their profile) with a particular context. Unlike the previous years where the participants were asked to build their own candidate list of suggestions, this year the organizers themselves released a fixed set of candidate suggestions for 272 contexts, each context representing a city in the United States. Participants were required to set up a server that could listen to requests from users and respond with relevant suggestions. For each request, which is a profile-context pairing, a ranked list of up to 50 ranked suggestions was to be returned.

- Input: Context including a location id i, latitude and longitude of the location associated with i, a user profile with a list of user preferences $\mathcal{P}(u)$ for a user u and additional information such as trip length and time of travel.
- Wanted: A ranked list of up to 50 suggested attractions to visit during the stay.

User profiles that are part of the input resquest correspond to the stated preferences of real individuals, primarily recruited through crowdsourcing sites, who also judge the returned suggestions. These users are asked to give a rating to the suggested points of interests. A rating could be any of the following: strongly interested (4), interested (3), neither interested nor uninterested (2), uninterested (1), strongly uninterested (0) or no rating (-1). P@5 is the main metric used for evaluation of the systems [1]. Mean reciprocal rank and modified version of time-based gain are other metrics used.

2 Setting Up the Server

To receive requests from users and to respond to these requests, a simple server was set up in python. To ensure that we could make changes to the server at any point during the live experiment without losing any requests we had two servers running with two different run IDs - WaterlooRunA and WaterlooRunB. However, the two runs had exactly similar scripts and the second one was created only for backup and for updating servers at any point if necessary.

[1] https://sites.google.com/site/treccontext/trec-2015/trec-2015-contextual-suggestion-track-guidelines

3 Gathering Data for Candidate Suggestions

The track organizers had released a total collection of approximately 1.2 million candidate points of interests for all the 272 context cities put together. Each candidate point of interest had an attraction ID, a context city ID, a title, and a URL. Collection of information for each of these candidate points of interest and techniques used for filtering out some of these candidates are discussed in this section.

3.1 Crawling Provided URLs

The home page of an attraction is expected to contain content that is highly relevant to the attraction. Further more, a large number of URLs from the provided collection were from standard tourist APIs such as Yelp[2], Foursquare[3] and Tripadvisor[4]. In such cases in particular, all the relevant information about the attraction is found at the given URL and any information on linked pages may not be relevant at all. Thus, we only gather content directly from the webpage pointed to by the provided URLs without downloading any content from linked pages. This downloaded content was used for extracting some key information (see Section 3.2) and also for filtering out some of the candidate suggestions (see Section 3.4).

3.2 Crawling Yelp, Foursquare and Tripadvisor

The majority of the URLs in the candidate suggestions belonged to either of these sources - Yelp, Foursquare or Tripadvisor. Web pages retrieved from these sources contain valuable information about each attraction such as a list of category, average user rating, tips, reviews etc. For each of the candidate attractions, we parsed the downloaded content to extract a set of category tags C_a and an average user rating r_a.

3.3 Additional Crawls

For those points of interests whose URLs were not from sources such as Yelp, Foursquare or Tripadvisor, we extracted relevant Foursquare URLs by searching against Bing[5]. The key phrase for this search included the title of the attraction followed by the keyword "Foursquare". Once the relevant Foursquare pages were retrieved, category tags C_a and average user rating r_a for each of the attractions were extracted (similar to Section 3.2).

3.4 Candidate Suggestions Filtering

Prior to the start of the live session, participants of the track were allowed to register their system for a test period to ensure that their system was working as required and to get some reliable feedback from the users to improve the design of the suggestion system if desired. From the responses that we received during the testing period, we had observed that a small number of the suggestions made by our system were rated -1. These suggestions mainly involved points of interests whose associated URLs were in Spanish, French and other non-English languages. These URLs and consequently the related points of interest were filtered out by using the python language detection package - langdetect[6]. Downloaded content from the URLs was used for this language detection, helping retain only those with English content.

The collection also included several duplicates. These were filtered out by searching for duplicate attraction titles and also duplicate URLs within the same context city.

[2] www.yelp.com
[3] www.foursquare.com
[4] www.tripadvisor.com
[5] https://www.bing.com/
[6] https://pypi.python.org/pypi/langdetect

4 Contextual Suggestions

Based on the information collected for each of the possible location IDs, the task requires us to construct a ranked list of attractions. For this, we designed a scoring function to quantify the likelihood that a specific user would rate a specific attraction highly and then ranked the candidates accordingly. We define our ranking in Section 4.1 and describe its offline and online computation components in Sections 4.2 and 4.3, respectively.

4.1 Ranking Definition

To compute a suitable ranking of the attractions for the suggestion, we use the following three pieces of information:

- the user's preferences $\mathcal{P}(u)$ given in the request, as well as
- the average online rating r_a and
- the set of category tags C_a associated with each attraction.

The ranking process consists of following two steps which we explain in more detail in the following subsections.

1. Construct a user profile for u from the list his/her preferences $\mathcal{P}(u)$.
2. Use the profile to compute a scoring function $\sigma(u, a, t)$ hat quantifies the likelihood of user u would rate a specific attraction highly and use it to construct a candidate ranking.

4.1.1 User Profile

Recall that every request contains user preferences in form of ratings (-1,0,1,2,3,4) for a subset of A. For each incoming request, we divide the set of all rated attractions into those like (rating 3 or 4) and disliked (rating less than 2) by the user and store their ids as $A^+(u)$ and $A^-(u)$, respectively. We define $C^+(u)$ ($C^-(u)$) as the set of all categories included in at least one of the attractions liked (disliked) by u. In other words:

$$C^+(u) := \bigcup_{a \in A^+(u)} C_a.$$

Liked categories $C^+(u)$ and disliked $C^-(u)$ together build the user profile.

4.1.2 Scoring Function and Ranking

We define $\sigma(u, a, a)$ s the number of category tags that are both considered to be descriptive of attraction a and liked by user u, i.e.:

$$\sigma(u, a) = |C^+(u) \cap C_a|$$

The output ranking consists of the top 50 elements in the list resulting from sorting all candidate attractions A_i for location id i lexicographically by $(\sigma(u, a), r_a)$.

4.2 Offline Computation: Data Preparation

The contextual suggestion task required us to respond to a request within 60 seconds. Due to this time constraint, we chose to implement a very simple ranking procedure and pre-processed the collected data sets as follows.

4.2.1 Fast Information Access

In order to have quick access to the information required for the ranking, we created the following two dictionaries and stored them in json format:

1. A dictionary containing all category tags C_a for each attraction a in A indexed by its id, and
2. a dictionary containing the average online user rating r_a for each attraction a in A indexed by its id.

4.2.2 Candidate Selection

The list of all possible attraction ids to choose from for the suggestions contains over 1.2 Million elements. As a result, online computation of the above defined score and ranking of all possible candidates was intractable. Instead, for every possible location id i, we pre-compiled the list A_i of all attractions associated with location i and restricted our attraction suggestions to this list. In addition, we pre-sorted the attractions a in A_i by their average rating r_a and stored the top 100 attractions in a sorted list called Ranked_Cand(i).

Note that Ranked_Cand(i) can be used as a default ranking for the suggestion. In fact, if the list of user prefences $\mathcal{P}(u)$ is empty, the desired ranking contains exactly the top 50 elements of Ranked_Cand(i) .

4.3 Online Computation: Personalized Ranking

The part of the ranking that cannot be computed in a pre-processing is the personalization component. When receiving a request including location id i and user id u, we computed the output ranking as follows.

1. We used the category dictionaries to Compute $C^+(u)$ for user u.
2. We computed $\sigma(u, a)$ as defined above for all candidates in A_i.
3. We sorted the pre-ranked set of candidates Ranked_Cand(i) by $\sigma(u, a)$, which gave use the desired lexicographical ordering.
4. Output the top 50 elements in the resulting ranking.

5 Ideas for the Future

In the previous sections we described the components implemented by our contextual suggestion system. We believe that the proposed ranking strategy suffices to produce adequate personalized recommendations in an online setting. However, there are many ways to improve the system. In the following list we propose possible improvements, which could be explored and incorporated in future work.

- In addition to the specific user profiles built for an online request, one could compute average user profiles based on data available from previously completed TREC runs. These profiles can be used to estimate the likelihood that an average person likes an attraction associated with a specific category. We recommend taking this likelihood into account when constructing contextual suggestions. As an example, consider the observation that most Americans like going to American Restaurants. Even if we do not know much about a specific user for which we want to make a suggestions (i.e. the user's preference list is sparse), then suggesting an American Restaurant is still likely going to be a good choice. On the other hand, if the categories associated with a certain attraction a are not very popular among an average user, then we should include a in the list of suggestions for user u only if the user's preferences reflect a liking for these categories. For instance, recommending a tatoo parlour to an average person might not be a good idea even if the tatoo parlour received exceptionally high online ratings.
- Another way to improve the ranking results could be to refine the primary scoring function $\sigma(u, a)$. So far we only compute the size of the overlap (#categories) between the categories liked by the user u and the categories associated with attraction a. Instead, one could explore the effectiveness of the following ranking method:

  1. For each category c, let $\ell(c, u)$ be the *number of times* that user u liked an attraction associated with c in their preferences, i.e. $\ell(c, u) := |\{a \in \mathcal{P}^+(u) : c \in C_a\}|$.
  2. Use $\bar{\sigma}(u, a) = \prod_{c \in C_a} \frac{\ell_c}{|\mathcal{P}^+(u)|}$ instead of $\sigma(u, a)$ as the primary score for the ranking.

  Note that there are many similar variants of this method that also take the frequency of category overlap into account, which might be worth exploring.

- One could also use additional context information such as the season in which the request is made to pre-filter the candidate lists. For example, a ski resort should only be suggested for a trip that takes place in the winter. Since this filtering requires the availability of additional information about when each attraction is available, we decided to leave this step as future work. Note however, that a subset of the data we crawled already includes this kind of information. We suggest incorporating these facts into the offline candidate selection step where possible.

References

[1] A. Dean-Hall, C. L. Clarke, J. Kamps, and P. Thomas. Evaluating contextual suggestion.

[2] A. Dean-Hall, C. L. A. Clarke, J. Kamps, P. Thomas, and E. M. Voorhees. Overview of the TREC 2012 contextual suggestion track. In E. M. Voorhees and L. P. Buckland, editors, *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012*, volume Special Publication 500-298. National Institute of Standards and Technology (NIST), 2012.

[3] A. Dean-Hall, C. L. A. Clarke, J. Kamps, P. Thomas, and E. M. Voorhees. Overview of the TREC 2014 contextual suggestion track. In E. M. Voorhees and A. Ellis, editors, *Proceedings of The Twenty-Third Text REtrieval Conference, TREC 2014, Gaithersburg, Maryland, USA, November 19-21, 2014*, volume Special Publication 500-308. National Institute of Standards and Technology (NIST), 2014.

[4] A. Dean-Hall, C. L. A. Clarke, N. Simone, J. Kamps, P. Thomas, and E. M. Voorhees. Overview of the TREC 2013 contextual suggestion track. In E. M. Voorhees, editor, *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*, volume Special Publication 500-302. National Institute of Standards and Technology (NIST), 2013.