Boosting Venue Page Rankings for Contextual Retrieval - Georgetown at TREC 2013 Contextual Suggestion Track

Jiyun Luo and Hui Yang
Department of Computer Science, Georgetown University
37th and O Street,NW, Washington DC, USA, 20057
jl1749@georgetown.edu,huiyang@cs.georgetown.edu

Abstract

We participate in the closed collection sub-track of the TREC 2013 Contextual Suggestion. The dataset that we use is an integrated collection of ClueWeb12 Category B, Wikitravel, and the city-specific sub-collection; all are from ClueWeb12. Since the Open Web is not used in our submissions, the task is essentially a retrieval task instead of a result merging task. Our system takes users' ratings of venues in a training city as inputs, and generates titles, document identification numbers, and descriptions for venues that fit users' interests in a new city. Ideal relevant documents for this task should be a list of Web pages each of which is a venue's homepage, which we call a "venue page". However, off-the-shelf search tools, such as Lemur, fail to retrieve such venue homepages from the collection. They either retrieve non-relevant documents or "yellow-page"-like pages that link to a long list of venue pages where the links are often broken and the destination pages are out of the collection. Therefore, large portions of the retrieved documents are not suitable as answers for contextual suggestion. To address this challenge, we experiment two different approaches, a precision-oriented approach and a recall-oriented approach, to boost the relevant venue pages' ranking.

1. Introduction

TREC 2013 Contextual Suggestion Track aims to provide travel suggestions in new cities for visitors based on their personal interests and ratings for venues in a city they have been to. The personal interests are provided by a survey of about 500 users' personal judgments towards fifty tourism suggestions in Philadelphia, PA. Users' judgments are scaled from 0 to 4, corresponding to "Strongly disinterested", "Disinterested", "Neutral", "Interested" and "Strongly interested".

Last year our submissions used the Open Web to find out the venues in the new cities. It was essentially a result merging task that utilizes a personalized learning to rank framework to merge top returned results from existing commercial search engines such as Google Place and Yelp. This year, we decide not to use the Open Web; instead, we concentrate on the core retrieval challenge posed in finding good documents in a closed collection, in this case, the ClueWeb12 collection.

The ideal relevant documents for this task should be a list of Web pages each of which is a venue's homepage. However, off-the-shelf search tools, such as Lemur, fail to retrieve such venue homepages from the collection. The following problems frequently occur: too many non-relevant documents, too many "yellow-page"-like pages that link to a long list of venue pages, the links are often broken, and the destination pages are out of the collection. These various forms of non-relevant documents and list pages are overwhelming. As a result, large portions of documents retrieved by off-the-shelf search tools are not suitable to be returned as answers for contextual suggestion. To address the above challenge, we experiment two different approaches, a precision-oriented approach and a recall-oriented approach, to boost the relevant venue pages' ranking.

The precision-oriented approach employs information extraction techniques to acquire venue names from Wikitravel then formulates structured queries to perform retrieval in the collection. We extract the names of venues, such as famous restaurants and shopping centers, from a given city's Wikitravel page. The venues are automatically extracted in sections named "See", "Do", "Eat", "Drink", and "Buy". After retrieval, we calculate the similarity between venue name and anchor texts in both incoming and current pages to locate the most relevant document within the closed collection for the venue. The description of a page is automatically extracted by a linguistic-based method. The recall-oriented approach divides all training venues into a two-level venue classification system, including landmark, amusement park, and Italian restaurant, then creates a representative language model for each category. The category-specific language models are used to perform retrieval for each individual category mentioned in a user's profile. Both approaches alleviate the problem of overwhelming non-relevant documents and list pages.

The remainder of this paper is organized as follows. Section 2 describes the dataset, data preparation and indexing. Section 3 presents the precision-oriented approach and Section 4 presents the recall-oriented approach. Section 5 shows how we reorganize the retrieval results based on user interests. Section 6 briefs description generation. Section 7 describes our submitted runs and result analysis.

2. Data Preparation and Indexing

The dataset that we use is an integrated collection of ClueWeb12 CatB, Wikitravel, and the city-specific sub-collection; all are from ClueWeb12. The integrated dataset is from following sources:

- **ClueWeb12 CatB**

The full ClueWeb12 dataset contains about eight hundred million Web pages. Pages are crawling from English-speaking countries' websites, twitter[1] and Wikitravel[2]. The ClueWeb12 CatB is used in our submission.

- **City-Annotated Sub-Collection**

This collection contains about thirty thousand documents. It is a sub-collection of ClueWeb12 which is distributed by the TREC organizers. Every document in this corpus has been marked with which city this website is about. All documents in this collection are tourism-related Web pages. We consider them as high-quality data. The weakness about this sub-collection is that it doesn't contain enough venues for each category, such as bar, restaurant or sightseeing.

- **Wikitravel**

Wikitravel is included in the ClueWeb12 collection. It is crawled from the English part of Wikitravel. A city's Wikitravel homepage is usually about that city's history, demographics, and what to do, to eat, to see, to drink in the city.

The dataset integration and data preparation is done in two steps. First, we merge the city-annotated sub-collection into ClueWeb12 CatB. We iterate each document in ClueWeb12 CatB. If a document's location information is provided in the city-annotated sub-collection, we add a "treccity" tag in that document and record which cities this document is about. For instance, document "clueweb12-1908wb-68-23037" is about Albany NY and Washington DC, we add the tag "<treccity>albany ny,washingtondc dc</treccity>" for it. Later on, during the retrieval phase, we conduct structured retrieval by utilizing this <treccity> field. The location information helps us to retrieve webpages in the city-annotated sub-collection in a higher priority than other ClueWeb12 documents. Second, we extract Wikitravel dataset out of ClueWeb12 and form a separate corpus. We extract famous entities of each city upon the Wikitravel dataset. By splitting Wikitravel to a separate corpus, we reduce the difficulty of retrieving famous venues from this corpus.

We adopt the Lemur Search Engine[3] to build index for the integrated ClueWeb12 CatB collection with stopword removal and Krovetz stemming [1]. To allow structured retrieval, we index the following fields: document number (docno), title, treccity and url. We use similar configuration to index the Wikitravel dataset.

3 Precision-Oriented Approach

3.1 Locating Wikitravel homepage for each city

For each city, we try to locate its Wikitravel page and extract venue names from this page. To do that, we issue a Boolean Lemur query formed by the city name and the full name of its state and perform retrieval against the Wikitravel index. This gives us a ranked list of Wikitravel pages for each city. For instance, for the city of Youngstown, OH, we issue a Boolean query "#band(Youngstown Ohio)". City Youngstown, OH's retrieval list is presented in Table 1.

Table 1 Retrieval List for Youngstown, OH

```
1. Northeast Ohio travel guide - Wikitravel
2. Youngstown travel guide - Wikitravel
3. Pages that link to Youngstown - Wikitravel
4. User:Cjensen/project/hotelmaker/Ohio – Wikitravel
5. Youngstown - Wikitravel
6. Talk:Northeast Ohio – Wikitravel …
```

Note that not all the top returned documents are relevant documents for our query. Moreover, we cannot rely on the first returned result to be relevant. Therefore, in order to locate one city's Wikitravel page, first we filter out Wiki User pages[4], Wiki Talk pages[5], Wiki Link pages[6] and Wiki

[1] https://twitter.com/
[2] http://Wikitravel.org/en/Main_Page
[3] http://www.lemurproject.org/
[4] http://en.wikipedia.org/wiki/Wikipedia:User_pages
[5] http://en.wikipedia.org/wiki/Help:Using_talk_pages
[6] http://en.wikipedia.org/wiki/Help:Link

Disambiguation pages[7] by filtering any page whose title are started with "user", "talk" or "page" or whose title contains word "disambiguation". Taking Table 1 as an example, the third, fourth and sixth retrieval results are filtered out first.

Figure 1 Examples of Wikitravel's Link, User, Talk and Disambiguation Pages

Next, we extract the city name and the state full name which are used in the Boolean query, and append a suffix "travel guide" to form a pattern p. E.g. for City Youngstown, OH, we get phrase "Youngstown Ohio travel guide". Finally we calculate the cosine similarity score [2] between the extracted phrase (p) and each retrieval document's title (t_j), and keep the document with the highest score as the Wikitravel page for that city. For City Youngstown, OH, its Wikitravel page is "2. Youngstown travel guide - Wikitravel".

$$\text{Wikitravel Page} = \text{the } i^{\text{th}} \text{ document, where } \cos(p, t_i) = \max_j(\cos(p, t_j))$$

3.2 Exacting venue names from Wikitravel

Table 2 The "See" section of document "Houma travel guide - Wikitravel"

Bayou Terrebonne Waterlife Museum [1] 7910 West Park Avenue, Tel 580-7200. Small museum focuses on local relationship to the wetlands habitat. Admission $3.
Chauvin Sculpture Garden, **Location** 5337 Bayouside Drive, Chauvin, LA, … The NSU Folk Art Studio is open from 1pm-3pm every Monday, Wednesday and Friday and by appointment. Free.   …
Southdown Plantation House [3] 1208 Museum Drive (just off Little Bayou Black Drive in the South west part of town) Tel 851-0154. Historic plantation house is now home to the Terrebonne Museum of History and Culture. Tues-Sat 10a-4p, $6
…

After retrieving one city's Wikitravel homepage, we examine the "See", "Do", "Eat", "Drink" and "Buy" sections in that page, and extract famous venues from these sections. For each section, first we extract all bold phrases. Usually these bold phrases are venue names. Second, we extract content within HTML lists, i.e., contents are tagged with <li> labels. If the content contains bold phrases, it means we already dealt this information once; hence we skip this content. Otherwise we split the content with comma, and keep the first part as a new venue. Finally we filter out venue names that contain only one term or more than 10 terms since they are more likely to be noise.

Table 2 shows that "Bayou Terrebonne Waterlife Museum", "Chauvin Sculpture Garden" and "Southdown Plantation House" are extracted as venue names, whereas "Location" is ignored.

[7] http://en.wikipedia.org/wiki/Wikipedia:Disambiguation

3.3 Retrieving ClueWeb12 documents for Wikitravel venues

After finding famous tour spots' names, we still need to find their most relevant document in ClueWeb12. The document must be within this closed collection. We generate a structured query in the following pattern and use it to search upon ClueWeb12 CatB:

- #weight(w1 #combine[title](venue name) w2 #combine(venue name) w3 #band(abbreviated location) w3 #band(full location))

where 'abbreviated location' uses the state abbreviation name, 'full location' uses the state full name. This is a weighted structured query. Keywords that occur in the title get w1 weights, and keywords that occur in the document get w2 weight. A document containing the correct location keywords, either in the abbreviation form or the full spelling, get w3 weight. We conduct several experiments to tune the weights, and empirically we set {w1,w2,w3 } as {0.5, 1.0, 0.2 }. Taking "Southdown Plantation House" as an example, the formulated query is:

- #weight(0.5 #combine[title](Southdown Plantation House) 1.0 #combine(Southdown Plantation House) 0.2 #band(houma la) 0.2 #band(houma louisiana))

This query expects that documents with "Southdown Plantation House" in the title and the body text and with "houma la" or "houma louisiana" in the content get the highest score.

After retrieval, we need to select one document that is most relevant to return as the answer for contextual suggestion. One approach is treating the document with the highest score as the Web page that best represents the query venue spot. Another more accurate approach makes use of anchor text. First, we issue the same query that we show in Section 3.3. Then, we extract all anchor texts and their outgoing urls for those anchor texts from the top five returned documents. Next, we compare the cosine similarity between the anchor text and the venue's name. Finally, we use the outgoing url of the anchor text which has the highest cosine similarity score to represent that venue. For instance, for "Southdown Plantation House", its highest scored anchor text and url pair is "southdown plantation house" and http://www.southdownmuseum.org/. We then need to map this url back to the ClueWeb12 collection to return the document id as the result.

4. Recall-Oriented Approach

In this approach, we perform contextual retrieval for a wide range of venues based on user interests and the city. We assume that users' information needs for traveling suggestions can be classified into 23 categories[8]. We search the fifty example suggestions in Yelp and use Yelp's category labels to determine the example suggestions' categories.

We adopt a two-level classification system to assign category labels to example suggestions. We use "top-level" to restaurants, whereas we assign more specific categories to example suggestions than "second-level" such as "Italian Restaurant" and "Turkish Restaurant" instead of labeling them all as "Restaurant". For other type of venues, we adopt "second-level" strategy to them, where category labels are more general.

We automatically generate a unigram language model for each category. We first generate some representative documents by querying the index using the category name, and then we take the words from the snippets in the retuned results, and retrieve any Wikipedia articles within them. From the snippets and Wikipedia articles we obtain top one hundred most frequent words, weighted by their counts. Table 3 lists a few example categories and their corresponding representative language model.

Table 3 Examples of categories and their representations

Category	Bag of Words
cafe	tea coffee cafe caffeine boiling Arabica chocolate cappuccino
landmark	landmarks historic buildings sites house monument structures preservation old ruin ancient architectural
performing art	arts performing plays theatre artists actors theater

We construct a Lemur structured query in the following format:

- #weight(w1 #combine(bag of words for one category) w2 #band(abbreviated location) w2 #band(full location))

[8] Italian restaurant, Turkish restaurant, Burmese restaurant, Greek Mediterranean restaurant, sandwiches restaurant, tour restaurant, restaurant, cafe, shopping, park, bar, brewery, landmark, amusement park, culture tour, zoo, performing art, spa, party event planning, game, tour, sport, museum

We set the parameters {w1, w2} as {1.0, 0.2} empirically. For instance, for City Houma, LA and Category "park", the query is:

✧ #weight(1.0 #combine(park landscape nature) 0.2 #band(houma la) 0.2 #band(houma louisiana))

Although we increase weights of a document if it contains the correct city name, the increment is lightweight. This is because that a great deal of list pages will contain many location terms in their pages, and will be ranked high. However, they are not our search targets. We eliminate the effects by assigning a light weight to city names and state names. However, all returned documents should be geographically relevant. Therefore, after a document list is retrieved, we post-process the returned document list and make sure that every document contains the correct location terms in it. The unmatched ones are removed.

5. Reorganizing the Retrieval Results

The retrieved results need to be merged according to each user's personal profile. We import all user ratings for the training city. It gives us a matrix that describes users' personal interests towards twenty three categories (Table 5). A mapping between user's degree of fondness and a numeric user interest weight is listed in Table 4.

Table 4 Degree of Fondness and User Interest Weight Mapping Table

fondness	strongly disinterested	disinterested	neutral	interested	strongly interested
weight	-0.75	-0.25	0	0.25	0.75

The mean fondness for a user and for a category are calculated as:

$$Mean(user_i) = \frac{\sum_{cat_j \in Categories} weight(user_i, cat_j)}{|Categories|}, \quad Mean(cat_j) = \frac{\sum_{user_i \in Users} weight(user_i, cat_j)}{|Users|}$$

If a user's fondness towards one category is bigger than the mean fondness of this user for all categories and also is bigger than the mean fondness received by this category from all users, we treat this category as this user's major interest. Besides major interests, other categories are considered as minor interests. For example, Profile 36's major interests include café and landmark, minor interests include bar.

Table 5 User-Interests Matrix

	cafe	bar	landmark	…	Mean of this user
Profile 35	0	-0.75	-0.75	-	-0.43
Profile 36	1.75	-0.5	4	-	0.89
…	-	-	-	-	-
Mean of this category	1.34	0.93	1.25	-	-

The result merging goes through several iterations. We assume that $user_i$ has major interests cat_1 to cat_n with weight w_1 to w_n and has minor interests cat_m to cat_{23} with weight w_m to w_{23}. $W_{all} = \sum_{i=1}^{23} w_i$, $W_{major} = \sum_{i=1}^{n} w_i$, |Major interests| = n, |Minor interests|=23-n. In an iteration, we generate contextual suggestions for categories ranked in descending order of category weight. In the first iteration, we only consider major interests and we output at least five suggestions. For cat_i, we output $\left\lceil \max(5, n) \times \frac{w_i}{W_{major}} \right\rceil$ pieces of cat_i type suggestions. For other iterations, we consider both major and minor interests. We output $\left\lceil 23 \times \frac{w_i}{W_{all}} \right\rceil$ pieces of contextual suggestions for cat_i. We stop the merging when we have 50 contextual suggestions.

6. Generating Descriptions

To generate the description for a suggestion, we examine a webpage's metadata for its description field. If a description is provided in the metadata, we use this description as our description; otherwise we extract the first five hundred and twelve characters from the body text and use them as our description. All HTML tags are removed.

7. Submissions and Results

We submit two runs to TREC 2013 Contextual Suggestion Track, namely BOW_V17 and BOW_V18. They are similar runs. The only difference is that they use different description generation methods. Our runs give the best performance among all approaches using the closed collection.

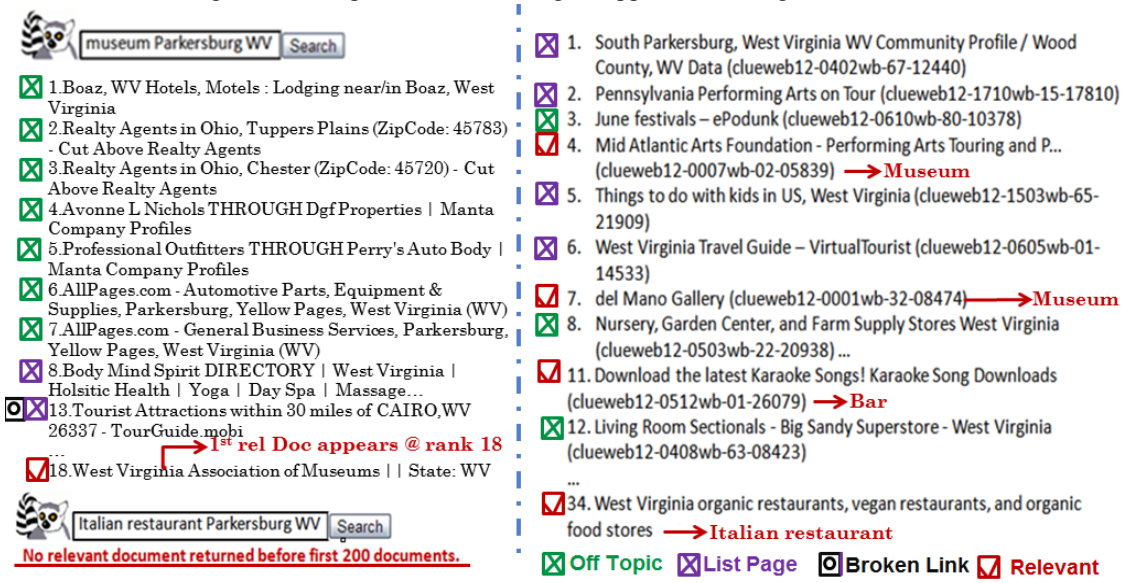

Figure 2 Lemur (left) vs. Our System (right)

Figure 2 shows a side-by-side comparison between the results retrieved by the default Lemur search engine and our system. The top documents retrieved by Lemur are either off-topic or list webpages. In our system, off-topic webpages rarely appear, and precision is much higher. Moreover, our retrieval results are well-balanced among various user interests in a personal profile.

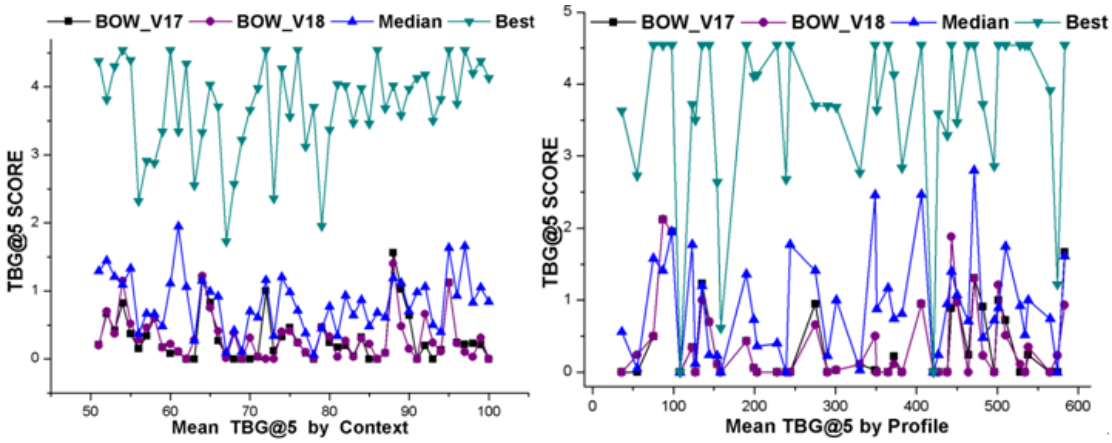

Figure 3 the perfomances of BOW_V17 and BOW_V18

Figure 3 shows the TBG@5 scores for our runs, the median and the best TREC runs. The median and the best runs are from the Open Web subtrack. Our runs' performance is roughly equal to the median Open Web run when we consider average score over different profiles. Our runs perform better on small cities (unpopular contexts) and worse on big cities (popular contexts) than the Open Web runs perform. This is because that an Open Web run can access many popular online search engines about venue suggestions; however, these search engines contain less useful information for unpopular venues.

Nonetheless, our approach investigates retrieval techniques that purely developed upon a closed collection and shows great potential for contextual retrieval.

8. References

1. R. Krovetz. Viewing morphology as an inference process. SIGIR 1993.
2. S. K. M. Wong, Wojciech Ziarko, and Patrick C. N. Wong. 1985. Generalized vector spaces model in information retrieval. SIGIR 1985.