Recommending Points-of-Interest via Weighted kNN, Rated Rocchio, and Borda Count Fusion [*]

Georgios Kalamatianos Avi Arampatzis

Database & Information Retrieval research unit,
Department of Electrical & Computer Engineering,
Democritus University of Thrace, Xanthi 67100, Greece.
`georkalamatianos@gmail.com, avi@ee.duth.gr`

Abstract

We present the work of the Democritus University of Thrace (DUTH) team in TREC's 2016 Contextual Suggestion Track. The goal of the Contextual Suggestion Track is to build a system capable of proposing venues which a user might be interested to visit, using any available contextual and personal information. First, we enrich the TREC-provided dataset by collecting more information on venues from web-services like Foursquare and Yelp. Then, we address the task with two different content-based methods, namely, a *Weighted kNN* classifier and a *Rated Rocchio* personalized query. Last, we also explore the use of a voting system, namely Borda Count, as a means of fusing the results of several suggestion systems. Our runs provided very good results, achieving top or near-top TREC performance.

1 Introduction

This paper presents the efforts of Democritus University of Thrace (DUTH) put at the TREC 2016 Contextual Suggestion Track. This year's Track consisted of two phases. We participated in the batch experiment (Phase 2—the main task of the Track in 2016), where we dealt with the task of re-ranking a set of candidate venues (Points-Of-Interest or POIs) for each request provided. In total, we were provided with:

- A complete collection of 1,235,844 POIs, together with their titles, corresponding URLs, and a web-crawl of the URLs.

[*]Proceedings of The Twenty-Fifth Text REtrieval Conference, TREC 2016, Gaithersburg, Maryland, USA, November 15–18, 2016. National Institute of Standards and Technology (NIST) Special Publication.

- A set of 442 requests made by 238 users. TREC eventually evaluated 58 requests made by 27 users.

Each provided request contained information about the user and the context in which the request was made. Specifically, each request was associated with:

- A user profile containing:
  - The user's age and gender (both optional).
  - A set of venues (either 30 or 60) that the user had rated at a scale of 0–4 (4=strongly interested, 3=interested, 2=neither interested or uninterested, 1=uninterested, 0=strongly uninterested) and also (optionally) assigned a set of tags from a controlled vocabulary (e.g. Bar-hopping, Desert, Live Music, etc.).[1]
- A context containing:
  - General location information about where the request was made (city, state, latitude and longitude).
  - The type of trip (business, holiday, other).
  - The intended duration of the trip (day trip, night out, weekend, longer).
  - The type of group that the user is traveling with (alone, friends, family, other).
  - The season in which the request was made (summer, winter, etc.).
- A set of candidates to re-rank, at the specified location of interest.

Taking all these under consideration, we were asked to re-rank the set of candidates to fit the user's preferences/profile and context. More information about the dataset and the Track's guidelines can be found online.[2]

We mainly focused on using the information from the users' preferences. First, we enriched the POI dataset with more information about each useful-for-the-experiments venue (i.e. all unique POIs in user-preferences and candidates) using both the TREC-provided web-crawl and the venue profiles we located in Foursquare[3] and Yelp[4] web services. We then built a suggestion model based on a *Weighted kNN* classifier, and another based on an adaptation of Rocchio's formula for relevance feedback, we call *Rated Rocchio*. Finally, we explored the use of the Borda Count voting system to combine the results of the two aforementioned models.

[1]The full set of possible tags: `https://github.com/akdh/csttools/blob/master/tags.csv`
[2]`https://sites.google.com/site/treccontext/trec-2016`
[3]`www.foursquare.com`
[4]`www.yelp.com`

2 Dataset Enrichment

Firstly, we identified the venue IDs that are useful to the experiments, namely, the set of IDs that constitute the preferences of the users and the candidate suggestions for all requests. We then proceeded to collect data about this set of venues using both the provided URLs and searching for the corresponding profiles in the web services Yelp and Foursquare. The TREC-provided URLs correspond to either the official web-page of the venue or its profile in a related web service. For each case, we proceeded in a different way.

2.1 URLs

For participants' convenience, TREC provided a set of web-crawls relieving us from performing this task ourselves. Instead, we processed the provided dataset, searching for the set of useful URLs we identified earlier. We then collected the meta-tags *description*, *title* and *keywords* for each POI. We ignored any other text present for two reasons. First, we found that a significant percentage of home-pages consist of some type of multimedia (thus also setting these venues at a disadvantageous point), and second, we found that home-pages that do contain text, contain an inherently uninformative content such as slogans, directions, etc.

2.2 Web-services

A significant percentage of the provided URLs correspond to a web-service profile, most of which are Yelp or Foursquare profiles. Some URLs are TripAdvisor links but the service explicitly prohibits the systematic data collection for research purposes, and the rest of the URLs which point to other similar services constitute a very small percentage so we treat them as simple URLs as explained above.

We attempted to match all venues to both Yelp and Foursquare profiles. Specifically:

- For URLs that corresponded to Yelp profiles, we used the exact coordinates provided by the web service's profile and the name of the venue, and submitted this information to Foursquare's search API to match the venue to a Foursquare profile. We then worked in the reverse way to match venues with Foursquare URLs to Yelp profiles.

- For the rest of the URLs, we attempted to match the venue to both Yelp and Foursquare profiles via the web services' search APIs using the name of the venue and the general location information.

For the venues that we successfully matched to any or both web-services' profiles, we collected and saved the complete profiles for later use.

Our final dataset of Foursquare and Yelp profiles is summarized in Table 1. We see that for more than 85% of the useful-to-the-experiments POIs (i.e. the

Table 1: *Web-service collection statistics*

Unique Candidates	18,752
Unique Preferences	60
Union of the above	18,808
Yelp profiles	15,295
Foursquare profiles	15,741
POIs with at least 1 of 2 web service profiles	16,680
POIs with both profiles	10,100

union of POIs in all user profiles and candidate POIs in all requests), we obtained at least a Yelp or Foursquare profile; that is a very good coverage. In the future, we will make these extra data available online, to accompany and enrich the TREC data.

3 Suggestion Processing

In order to provide suggestions for each request, we revisited and revised the two methods we first used in DUTH's TREC 2013 participation [2], which were also used in [3, 4], while we also attempted to combine their results using a voting system.

3.1 Suggestion Model based on a Weighted kNN Classifier

Based on the kNN Classification method [1], we attempted to predict a user rating for each candidate venue based on the actual user ratings of the k neighbors semantically nearest to the candidate. We then ranked the candidate venues based on the predicted rating. Specifically, we proceeded as following:

- *Indexing*: We generated an index per user, containing the data collected for each user preference. This data consisted of a bag-of-words containing the metadata (*description*, *title*, *keywords*), the Foursquare profile data (*description*, *title*, *tags*, *phrases*), and the Yelp profile data (*description*, *category*, *title*). For indexing, we used Indri version 2.2[5] with default settings, except that we enabled the Krovetz stemmer, as suggested in [3].

- *Query Generation*: We generated a query for each candidate venue which consisted of a bag-of-words containing the meta-data, the Foursquare profile, and the Yelp profile data, as used in the index.

- *Rating Candidates*: For each candidate venue, we submitted its generated query to the corresponding user index which returned the list of the user's preferences scored for their semantical similarity to the candidate venue.

[5] `www.lemurproject.org`

To compute a rating p for the candidate, we used a weighted average of the nearest neighbor ratings as proposed in [2]:

$$p = \frac{\sum_{i=1}^{k} s_i R_i}{\sum_{i=1}^{k} s_i}\,, \tag{1}$$

where k is the number of the examined nearest neighbors, s_i is the tf-idf similarity value between the candidate and its i-th neighbor (we enabled Indri's tf-idf retrieval model for that), and R_i is the user's rating for the i-th neighbor.

We used $k = 7$, since for some candidates Indri returned no more than 7 similar venues. This is because Indri does not return listings that bare no similarity (i.e. no common keyword) to the query.

3.2 Suggestion Model based on a Rated Rocchio Method

In this model, we attempted to create a personalized query per user. This query consisted of terms weighted via a custom version of Rocchio's formula usually employed for relevance feedback [6]. The model is summarized in the following steps.

- *Indexing*: We generated an index per context, containing all available POIs in the area of interest (i.e. the general location where the request was made). As in the kNN-based model, these data consisted of a bag-of-words containing the metadata (*description*, *title*, *keywords*), the Foursquare profile data (*description*, *title*, *tags*, *phrases*), and the Yelp profile data (*description*, *category*, *title*).

- *Query Generation*: Let $D_i =< d_{i,1}, d_{i,2}, ..., d_{i,M}, >$ be a training example (a rated venue), where M is the number of terms in the training set of all preferences of a user. The $d_{i,j}$ is the weight of j term in the D_i. The term weight was calculated as $d_{i,j} = 1 + log(f_{i,j})$ where $f_{i,j}$ is the frequency of appearance of the j-th term in D_i. We used a tf-only weight as the inclusion of an idf term did not improve our preliminary experiments with last year's dataset. We then used the following formula to calculate the personalized user query $\vec{Q}$, where R_j is the set of vectors (training examples) of the user's preferences rated as j:

$$\vec{Q} = \sum_{j=0}^{4} \left((j-2) \frac{1}{|R_j|} \sum_{\vec{D} \in R_j} \vec{D} \right). \tag{2}$$

  In this way, we regarded ratings of $j = 2$ as neutral, 3 and 4 as positive, and 0 and 1 as negative.

  Finally, we generated the query by creating a list of the terms together with their calculated weights. For example, using the Indri Query Language notation:

#weight(1.5 *restaurant* 1.2 *steak* ... 0.2 *good*)

  We also applied a cut-off threshold of 20 highest weighted terms for each query to achieve a more precise query.

- *Rating Candidates*: We submitted the generated query to the context index, that is, the index containing all the POIs in a context that the user is interested in. From the returned list of venues, ranked by similarity to the query (here, we used Indri's default Language Model for retrieval/ranking), we filtered out the venues that did not belong to the set of candidates of the request, and suggested the rest in their ranked order.

Note that while the Weighted kNN method of Section 3.1 predicts ratings, the Rated Rocchio method as defined here does not predict ratings but ranks the candidates—there are no guarantees that even the first ranked POI would correspond to a higher-than-neutral rating.

3.3 Suggestion Fusion via Borda Count Election

We investigated the use of election methods as a means to fuse the results produced by the two separate suggestion methods. We employed the *Borda Count* election method, which originates from social theory in voting and has been previously used as a fusion method of separate retrieval systems, e.g. [5]. For that purpose, we used the lists of suggestions as ballots and each model as a voter.

Specifically, for each suggestion of rank r, we assigned a score $p = n - r$ where n is the number of candidates. Although other formulas such as $p = n - (r - 1)$, and $p = \frac{1}{r}$ are frequently used, $p = n - r$ provided the best results in preliminary experiments with last year's dataset. We then added the p scores of each candidate for both methods and re-ranked the list of suggestions based on the final score.

We can summarize the function of the Borda Count method in the following formula:

$$s_i = \sum_{j=1}^{m} (n - r_{i,j}) \,, \tag{3}$$

where s_i is the final score of candidate i, m is the number of combined voters (two in our case), $r_{i,j}$ is the rank of candidate i according the j voter/method, and n is the number of candidates.

4 Experiments & Results

In this section, we present the experimental evaluation of our proposed methods. First, we present the official evaluation measures, and then our official runs and results. Some extra runs are also presented, since we found a bug in our system which influenced two out of our three officially submitted runs.

4.1 Evaluation Measures

The suggestions were evaluated according to the NDCG@5, P@5, and MRR measures, macro-averaged over all the requests. The definitions of these measures are:

- *Normalized Discounted Cumulative Gain at 5 (NDCG@5):* A measure that employs a gain factor to take into account the position in which each relevant suggestion was returned, as well as its relevance score (rating) according to the qrel file (ground truth).
- *Precision at Rank 5 (P@5):* The fraction of relevant suggestions within the top-5 results.
- *Mean Reciprocal Rank (MRR):* Average, inverse rank of the first relevant suggestion.

Note that all the above measures are sensitive to early precision, due to the narrow rank cut-offs (5) in NDCG and Precision, and the rank position of only the first relevant result in MRR. Consequently, they model a user with a mobile device, with potentially limited screen space where only five suggestions can be displayed simultaneously (NDCG@5, P@5), or with available time to visit only single POI (MRR).

4.2 Official Runs & Extras

In this year's Track, 13 teams participated in Phase 2 experiments with a total of 30 runs (including our team). We submitted three official runs:

- `DUTH_knn`, with the Weighted kNN method described in Section 3.1.
- `DUTH_rocchio`, with the Rated Rocchio method described in Section 3.2.
- `DUTH_bcf`, a fusion of the two previous runs with the Borda Count method described in Section 3.3.

Unfortunately, we later found a bug in `DUTH_knn` which hurt its effectiveness; this also invalidates the fused `DUTH_bcf` run. The results of our official runs, as well as the de-bugged ones (with an asterisk), are presented in Table 2.

Our best performing official run is `DUTH_rocchio`, which came up as the 1st best run in NDCG@5 of all participants, while it also achieved the 2nd and 4th positions in MRR and P@5, respectively. Unfortunately for us, the bug we found in the official `DUTH_knn` run was proved too costly; the de-bugged `DUTH_knn*` run surpasses `DUTH_rocchio` in NDCG@5, so it would have been another 1st best. Also, its P@5 and MRR are considerably better, now comparable to `DUTH_rocchio`.

The official `DUTH_bcf` is invalid, since it is based on the buggy `DUTH_knn`; consequently, we see no value commenting on it. Concerning the de-bugged `DUTH_bcf*`, which fuses `DUTH_knn*` and `DUTH_rocchio`, it achieves a slightly

Table 2: *Effectiveness of official and de-bugged (with an asterisk *) runs, and official run rank positions (in parentheses) within the 30 submitted runs by all participants. Best effectiveness per measure is in bold typeface.*

	NDCG@5	P@5	MRR
`DUTH_knn`	.3116 (7)	.4345 (8)	.6131 (10)
`DUTH_knn*`	**.3388**	.4690	.6697
`DUTH_rocchio`	.3306 (1)	**.4724** (4)	**.6801** (2)
`DUTH_bcf`	.3259 (4)	**.4724** (4)	.5971 (13)
`DUTH_bcf*`	.3232	.4552	.6165

worse performance in NDCG@5 and P@5 than both runs it combines, and worse in MRR. Thus, ranked-based fusion does not seem like a promising method in this dataset and measures, although our preliminary experiments with last year's data had shown otherwise.

The TREC official evaluation provided also some additional measures beyond the above three. In these additional measures, `DUTH_rocchio` achieved the 1st best NDCG, 5th MAP, 4th bpref, 2nd P@10, and 3nd Rprec. Thus, the Rated Rocchio method can also be effective under a variety of use cases beyond early-precision mobile applications. Ironically, the invalid `DUTH_bcf` achieved the 1st best MAP, bpref, and Rprec, which may indicate that ranked-based fusion may be more suitable for use cases not too sensitive to early precision. We have not yet evaluated the de-bugged `DUTH_knn*` and valid `DUTH_bcf*` using the additional measures.

5 Conclusions

In this year's Track, we further developed and built upon the two methods we first presented in Contextual Suggestion 2013. Specifically, we fine-tuned them using last year's data (2015), and made an attempt to combine/fuse their results as a third method.

The first method was based on a Weighted kNN classifier. We issued a candidate venue as a query to an index of all user's rated venues, and predicted its rating by taking the weighted average of the ratings for the 7 most semantically similar rated venues, weighted by their tf-idf score. The second method was based on a Rocchio-like method we developed for multi-graded relevance environments, called Rated Rocchio. Using a user's rated venues as training examples, we built a personalized query for the user. Specifically, we built a centroid per rating and combined/added those using their corresponding ratings as contributing factors, offset by 2 so as ratings 0 and 1 provided negative feedback with -2 and -1 weights, respectively. Rating 2 was eliminated as neutral. The Rated Rocchio query was run on an index with the POIs in the area of interest in order to rank the candidates. The third method fused the results produced by the first two suggestion methods. We employed the Borda Count election method, which originates from social theory in voting. In summary, we are satisfied again with our performance this year, since our runs achieved top or near-top TREC effectiveness.

In the future, further improvements could come from the following. In the Weighted kNN method, the tf-idf score, which is suitable for ranking, may not be the best choice as a semantical distance measure/weight—in this respect, other mechanisms could be explored. In the Rated Rocchio method, we assumed a linear behavior in user's ratings, e.g. a rating of 4 is taken as two times more relevant than a rating of 3, etc. Furthermore, we discarded all neutral ratings of 2, which may have helped with Recall. In this respect, alternative centroid weightings could be investigated. Combining several methods also merits further investigation. A procedure which selectively combines only good individual performances but falls back to an individual run otherwise, seems to make sense. Here, the query performance prediction literature may be of use. Additionally, note that the main reason we resorted to a ranked-based fusion method is that the Rated Rocchio method does not predict ratings (in contrast to Weighted kNN) but ranks the candidates. In this respect, ways that map the Rocchio retrieval scores to ratings could be investigated, so as to enable the use of rating-based fusion by e.g. taking the average rating across systems.

Acknowledgments

The authors would like to thank Hadi Hashemi, PhD candidate at University of Amsterdam (UvA) and organizer of the Track, for providing some more insight and statistics on the dataset.

References

[1] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992. doi: 10.1080/00031305.1992.10475879. URL `http://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879`.

[2] George Drosatos, Giorgos Stamatelatos, Avi Arampatzis, and Pavlos S. Efraimidis. DUTH at TREC 2013 contextual suggestion track. In Ellen 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. URL `http://trec.nist.gov/pubs/trec22/papers/DUTH-context.pdf`.

[3] George Drosatos, Pavlos S. Efraimidis, Avi Arampatzis, Giorgos Stamatelatos, and Ioannis N. Athanasiadis. Pythia: A privacy-enhanced personalized contextual suggestion system for tourism. In *39th Annual IEEE Computers, Software and Applications Conference (COMPSAC 2015)*, pages 822–827. IEEE Computer Society, july 2015. doi: 10.1109/COMPSAC.2015.88.

[4] Pavlos S. Efraimidis, George Drosatos, Avi Arampatzis, Giorgos Stamatelatos, and Ioannis N. Athanasiadis. A privacy-by-design contextual suggestion system for tourism. *Journal of Sensor and Actuator Networks*, 5 (2):10, 2016. ISSN 2224-2708. doi: 10.3390/jsan5020010. URL `http://www.mdpi.com/2224-2708/5/2/10`.

[5] Rabia Nuray and Fazli Can. Automatic ranking of information retrieval systems using data fusion. *Inf. Process. Manage.*, 42(3):595–614, 2006. doi: 10.1016/j.ipm.2005.03.023. URL `http://dx.doi.org/10.1016/j.ipm.2005.03.023`.

[6] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart retrieval system - experiments in automatic document processing*, pages 313–323. Englewood Cliffs, NJ: Prentice-Hall, 1971.