DUTH at TREC 2013 Contextual Suggestion Track*

George Drosatos[†1,2], Giorgos Stamatelatos[1,2],
Avi Arampatzis[1,2] and Pavlos S. Efraimidis[1,2]

[1]Electrical & Computer Engineering Dept., Democritus University of Thrace
[2]Institute for Language & Speech Processing, Athena Research & Innovation Center
University Campus, Xanthi 67 100, Greece
{gdrosato,gstamat,avi,pefraimi}@ee.duth.gr

Abstract

In this report we give an overview of our participation in the TREC 2013 Contextual Suggestion Track. We present an approach for context processing that comprises a newly designed and fine-tuned POI (Point Of Interest) data collection technique, a crowdsourcing approach to speed up data collection and two radically different approaches for suggestion processing (a k-NN based and a Rocchio-like). In the context processing, we collect POIs from three popular place search engines, Google Places, Foursquare and Yelp. The collected POIs are enriched by adding snippets from the Google and Bing search engines using crowdsourcing techniques. In the suggestion processing, we propose two methods. The first submits each candidate place as a query to an index of a user's rated examples and scores it based on the top-k results. The second method is based on Rocchio's algorithm and uses the rated examples per user profile to generate a personal query which is then submitted to an index of all candidate places. The track evaluation shows that both approaches are working well; especially the Rocchio-like approach is the most promising since it scores almost firmly above the median system and achieves the best system result in almost half of the judged context-profile pairs. In the final TREC system rankings, we are the 2nd best group in MRR and TBG, and 3rd best group in P@5, out of 15 groups in the category we participated.

1 Introduction

TREC 2013 is the second year that the Contextual Suggestion Track is running. The track's goal is to investigate search techniques considering as context only the user's location, as well as, user interests via personal preferences and past history. In other words, the track focuses on one situation: a user with a mobile device with limited interaction but some sort of a user profile; who is in a strange town; and who is looking for something to do. There is no explicit query; the implicit query is: *Here I am, what should I do?*

The remainder of this report is organized as follows. Section 2 describes our methodology for the context processing. The proposed suggestion models with details about the steps that we

*In: Proceedings of the Text REtrieval Conference (TREC), NIST, 2013.
[†]corresponding author

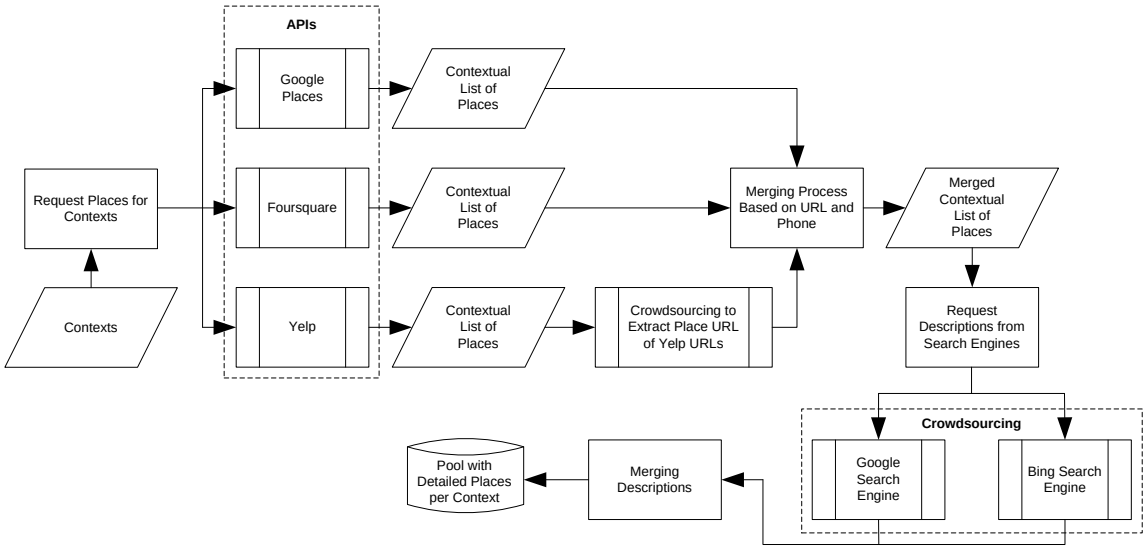


Figure 1: A system flowchart of the context processing.

followed are presented in Section 3. Our submitted runs and the official results of TREC Contextual Suggestion Track are described in Section 4. Finally, Section 5 draws our conclusions.

2 Context Processing

The goal of the Contextual Suggestion Track 2013 was to recommend interesting places and activities from either the open web or ClueWeb12. In our case, we used the open web because we did not obtain the ClueWeb12 dataset in time. Specifically, we followed a context processing approach similar to [4] but we enriched it with extra place search engines that provide POIs (Points of Interest) based on geographical location and a range of place types. Figure 1 shows an overview of our context processing. In more detail, the steps that we followed to create a pool of POIs for every context (i.e. a primary city of 50 randomly selected metropolitan areas) are:

1. **POI collection:** We generate appropriate queries for every context and submit them to three place search engines, namely, *Google Places*[1], *Foursquare*[2] and *Yelp*[3]. The submitted queries have as parameters the geographical location (latitude and longitude) of the context and a set of place types that are expected to be interesting to the user. Several of the place types covered by the search engines seemed to be irrelevant to the requirements of the TREC challenge. Thus, we used the description of the TREC challenge and the example data sets to define the set of place types that are relevant to the specific search task. Each search engine has its own vocabulary for place types and consequently we selected three sets of place types, one for each search engine. The sets of place types for every engine are:

[1] `https://developers.google.com/places/`
[2] `https://foursquare.com`
[3] `http://www.yelp.com`

- ***Google Places:*** Amusement Park, Aquarium, Art Gallery, Bar, Book Store, Bowling Alley, Cafe, Movie Theater, Museum, Park, Restaurant, Shopping Mall, Zoo, Grocery Store/Supermarket, Casino, Night Club, Beauty Salon, Travel Agency, Jewelry Store, Library, Church.
- ***Foursquare:*** Arts & Entertainment, Outdoors & Recreation, Food, Farmers Market, Smoke Shop, Mall, Gourmet Shop, Nightlife Spot, Spa/Massage, Gift Shop, Gym/Fitness Center, Monument/Landmark, Tourist Information Center, Library, Spiritual Center, Jewelry Store.
- ***Yelp:*** Arts & Entertainment, Landmarks & Historical Buildings, Food, Tobacco Shops, Shopping Centers, Party & Event Planning, Tours, Nightlife, Active Life, Restaurants, Beauty & Spas, Cards & Stationery, Travel Services, Department Stores, Religious Organizations, Jewelry, Libraries.

To overcome the limits on the number of results returned by the APIs of the engines (*Google Places:* 200 results per query, *Foursquare:* 50 results per query, *Yelp:* 20 results per query), we split each query into subqueries with different geographic coordinates to retrieve more results. The total area that we cover with the above search engines is a $6\,\mathrm{km}^2$ square per city. More precisely, in Foursquare and Yelp we use bounding boxes of $400\,\mathrm{m}^2$ with different geographic coordinates, and in Google Places we use circular areas of radius 849 m and different geographic coordinates as center. Note that a square of side 1200 m can be inscribed into a circle of radius 849 m. Consequently, we submit $5 \times 5 = 25$ queries to Google Places and $15 \times 15 = 225$ queries to Yelp and Foursqaure to cover the $6\,\mathrm{km}^2$ area. We use wider (i.e. covering larger areas) subqueries in Google Places since the corresponding API returns more results than the other search engines.

2. **Obtaining the URLs:** The Yelp API provides only the Yelp URLs. The actual URLs of places are in the contextual list of places of Yelp; in order to retrieve them from there we use a crowdsourced distributed technique (described in Section 2.1) due to high volume of retrieval operartions that have to be executed.

3. **Raw POI merging:** The partial contextual lists of places of every search engine are joined into a single one. Two items in different lists are merged if their URLs or their phone numbers are identical, and the distance between their titles is smaller than a threshold (< 0.1). For the computation of distance between the titles, we use the Jaro-Winkler distance [10] that is a measure of similarity between two strings. The general rules that we follow when two places match with the one of two previous ways are:

   - When two matched items differ in their URL, we keep the URL of first engine in the following order of decreasing priority: Google Places, Foursquare, Yelp.
   - When two matched items differ in their title, we keep the longest title.
   - The set of place types of the merged item is the union of the place types of the individual items, that is, we keep all the place types assigned by the search engines.

   The merged list of places contains only places that have a website URL in accordance to the requirements of the track [9]. In Table 1, we show aggregated data about the list of collected places for each context.

Table 1: Statistical information about the contextual list of places.

Context	Google	Foursquare	Yelp	Merged / Sum
Crestview, FL	103	33	38	131 / 174
Anniston, AL	139	53	26	168 / 218
Sumter, SC	147	52	40	173 / 239
...	...	...	...	...
Orlando, FL	590	328	497	1008 / 1415
Atlanta, GA	694	559	738	1378 / 1991
Washington, DC	812	1126	1275	2378 / 3213
Total *(with URLs)*	14945	7664	8394	22600 / 31003
Total *(retrieved)*	—	68517	15787	—

Sorting by places ↓

4. **Enrichment of POI descriptions:** The final step is to enrich the merged list of places with descriptions. We use as descriptions the snippets returned by general purpose web search engines when the URL of a place is submitted as a search query. In particular, we use Google[4] and Bing[5] to collect the corresponding snippets. For retrieving the snippets we use a crowdsourcing mechanism (described in detail in Section 2.1) mainly in order to overcome the querying-limits of engines. The snippets are used to generate two versions of the description for each place. The first version, or else the "retrieval" version, of the description is used for the retrieval tasks in our contextual suggestion algorithms. The second version, or else the "presentation" version, is the description presented to the users. In merging the snippets, for the retrieval descriptions we keep only the snippet of Google (or Bing's when Google fails to retrieve any result), and for the presentation descriptions, we keep both of them if the Jaro-Winkler distance [10] of snippets is bigger than a threshold (≥ 0.25), otherwise we keep the snippet of Google (or Bing's when Google fails).

2.1 Crowdsourcing mechanism

The crowdsourcing mechanism we used to collect the necessary snippets and website URLs from Yelp is based on a distributed HTTP client-server architecture in order to overcome certain search engine limitations and exploit modern web browser heuristics on malformed HTML translation. Our client is executed in a web browser (Google Chrome) extension context, as a silent background process without the need of user interaction. At regular intervals, each client requests a new query from the service, submits it to the engines (Google, Bing or Yelp) and reports back the snippet of the most relevant result or the URL in the case of Yelp. The service prepares the queries to be submitted to the engines, distributes the workload among clients and uses a MySQL database backend to store the information retrieved by the clients. A small community of volunteers installed our client to their Chrome browsers. Overall, an average of about 25 instances of the client were used to collect the snippets of 19,825 unique URLs and the website URLs of 15,787 Yelp places, 53% of which included a website. Still, the process lasted around 20 days.

[4] `http://www.google.com`
[5] `http://www.bing.com`

3 Suggestion Processing

In this section we present two suggestion models that are applied to the pool of POIs that are collected in the context processing (Section 2). The models are depicted in Figure 2.

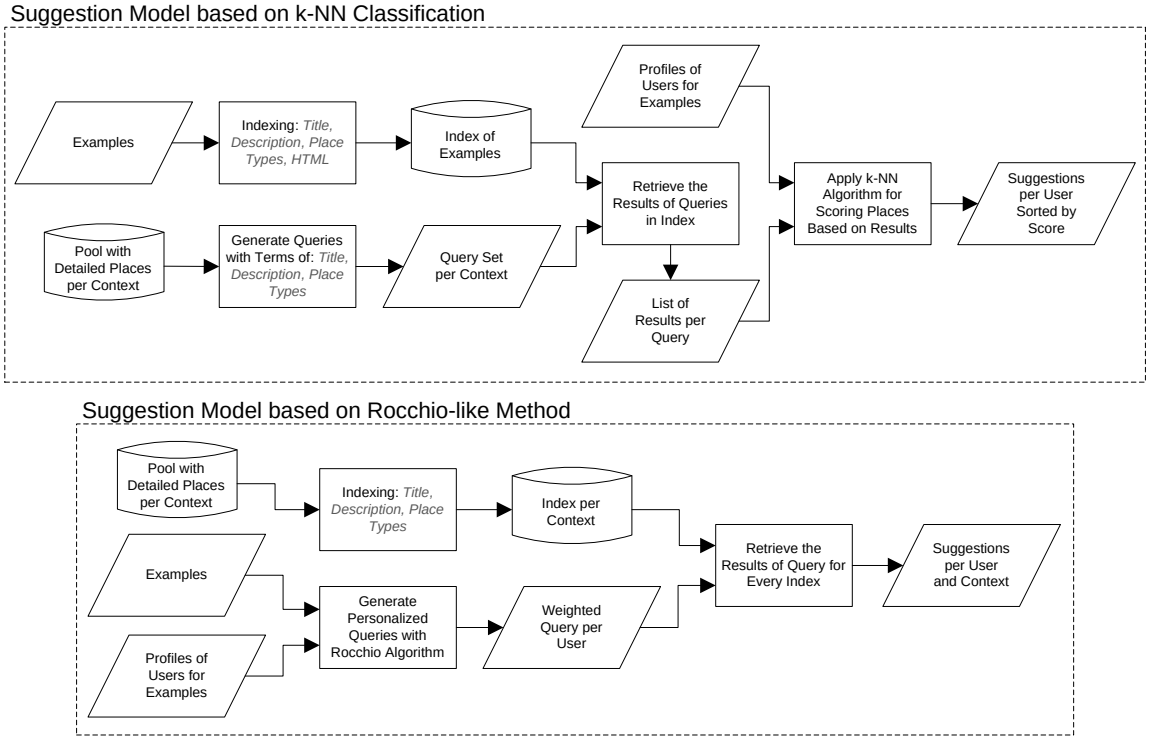


Figure 2: An overview of the proposed suggestion models.

3.1 Suggestion Model based on k-NN Classification

The idea is to assign a rating or score to each candidate POI based on the ratings of its k semantically nearest POIs (neighbors) in the user profile. Then all candidate POIs are ranked in a decreasing order of their assigned scores.

The model is implemented in three main steps:

1. *Indexing the rated POIs.* In order to be able to find the k semantically nearest (rated) neighbor POIs of a candidate (unrated) POI, we create an index of the POIs that are part of the user profiles and have been evaluated and rated by the users. For each rated POI we index its title, description, place types and the text of its website. The place types of the rated POIs are not provided by the track, but we retrieve them from the three place search engines that we use in context processing (as we described in Section 2). For indexing, we use Indri[6] v5.5 with the default settings of this version, except that we enable the Krovetz stemmer [5].

[6] `http://www.lemurproject.org/`

2. *Generating queries from candidate POIs.* We generate a query per candidate POI in a context. The query consists of the POI title, place types and the description of the POI that we retrieved in the context processing. From the query, we remove all punctuation and special characters.

3. *Scoring candidate POIs based on their k-NNs.* We submit the queries (per context) that are generated in Step 2 to the index that is created in Step 1 in order to rank the rated POIs in an increasing semantical distance. In a standard k-NN [1, 6], a candidate POI (represented by its corresponding generated query) would be assigned the majority rating of the top-k retrieved POIs. In initial experiments, however, we found that taking into account the ranks or retrieval scores of the top-k results is beneficial. We experimented with several formulas using cross-validation, such as linear (e.g. Borda Count) or exponential weights decreasing with the rank, and we settled for the following best-performing formula for scoring each candidate POI P:

$$P = \frac{\sum_{i=1}^{k} s_i \cdot R_i}{\sum_{i=1}^{k} s_i}, \quad R_i = \frac{R_i^D + R_i^W}{2}, \tag{1}$$

where s_i is the Indri tf-idf score of the ith ranked POI. This formula assigns to a candidate POI a score equal to the weighted average of the ratings of the k-nearest-neighbor POIs in a user profile, where weights are given by tf-idf similarity. As POI's rating R_i we use the average rating of the description (R_i^D) and the website (R_i^W), because in Step 1 we index both the description and the text of website. The value of k that we use in our suggestions was optimized to $k = 23$ by using 5-folds cross-validation [8] on the example places. The scored candidate places are then ranked in a decreasing order of their scores.

3.2 Suggestion Model Based on a Rocchio-like Method

The idea is to use the rated POIs in the user profile to generate a query using a Rocchio-like relevance feedback method [7, 6]. Then the generated query is used to score and rank all candidate POIs

The model is implemented in three main steps:

1. *Indexing all candidate POIs per context.* Per context, we build an index of the POIs collected during the context processing. For indexing, we use the title, description and the place types. As noted earlier, the description used for the index is only the snippet of Google, or Bing when Google failed to retrieve any result. The POIs are indexed with the Indri v5.5, using the default settings of this version, except that we enable the Krovetz stemmer [5].

2. *Generating a personalized weighted query per user with a Rocchio-like relevance feedback method.* We generate a query per user, representing her preferences, based on her rated POIs. The terms used in the query are taken from the title, the place types and the descriptions of the rated POIs. Let $D_i =< d_{i,1}, \cdots, d_{i,M} >$ be a training example, where M is the number of terms in the training set of all examples. The $d_{i,j}$ is the weight of j term in the D_i; we use tf-only logarithmic weighting: $d_{i,j} = log(1 + f_{i,j})$, where the $f_{i,j}$ is the frequency of j term in the D_i. Then, the trained weighted query of user u is given by the equation:

$$Q_u = \sum_{j=0}^{4} \left((j-2) \frac{1}{|R_{j,u}|} \sum_{D \in R_{j,u}} D \right) \quad (2)$$

where the $R_{j,u}$ is the subset of the examples that were rated by user u with score j. In other words, we take the centroids per rating j, multiply them with the normalized rating $j-2$ (so that the neutral rating's centroid, i.e., for $j=2$, does not contribute anything—it is zeroed), and then add the centroids in a Rocchio relevance feedback fashion. All the terms of Q_u that have weight less or equal than zero are excluded from the query. The weight of every term is included in the query by using the Indri Query Language and has, e.g., the following form:

```
#weight( 3.0 museum 2.7 art ··· 0.1 nice )
```

3. *Retrieving suggestions.* We submit the personalized queries of the users that are generated in the second step in the index of every context (created in the first step). Our search engine is again the Indri v5.5 with the default (LM) retrieval model. The results of each query, with a possible cutoff threshold (e.g. top-50 results), are our suggestions for the corresponding user and context.

3.3 An Attempt to Evaluate via a User Study

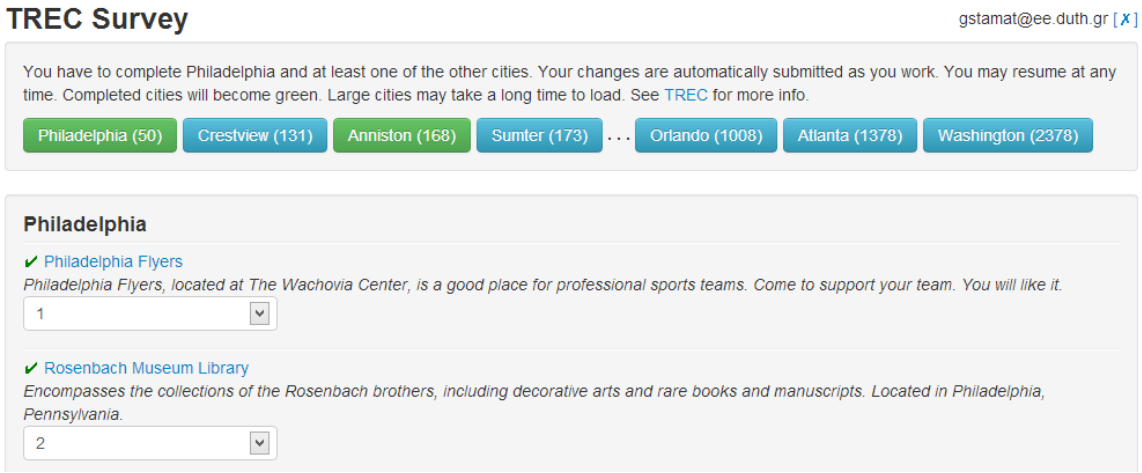


Figure 3: Screenshot of a user who has completed the rating of Anniston and the 50 examples from Philadelphia.

In order to evaluate our methods, we developed an online tool which prompts the user to rate the description of a POI according to her own interests with a scale of 0 to 4. On the main modal of the application, the user is presented with the list of POIs that we collected, each of which contains its name and the associated description (Section 2, Step 4). Initially, the user is asked to rate the 50 example places from Philadelphia, essentially creating her profile. She also chooses a city

Table 2: Mean of results over all the profiles and contexts for P@5, MRR and TBG measures.

	P@5	MRR	TBG
Runs:			
DuTH_A	0.3283	0.4836	1.3109
DuTH_B	0.4090	0.5955	1.8508
Difference:			
DuTH_B vs _A	+24,58%	+23,14%	+41,19%

and rates its consisting POIs using the same criteria. We then feed this profile to our models and compare the suggestions to the actual ratings that the user provided using the Pearson product-moment correlation coefficient [3]. However, due to the low number of participants (specifically 5) we managed to involve before the submission deadline, this method did not prove particularly useful. Thus, we leaned towards the use of cross-validation described in Section 3.1 (Step 3).

4 Runs and Results

We submitted two runs to the TREC 2013 Contextual Suggestion Track. The first run is labeled DuTH_A and uses the k-NN classification technique, while the second run (DuTH_B) is based on the Rocchio-like approach (Section 3).

The evaluation results according to the P@5, MRR and TBG measures over all the profiles and contexts of our two runs are reported in Table 2. The definitions of these measures are:

- *Precision at Rank 5 (P@5):* The fraction of suggestions within the top-5 results where the user liked both the description and the geographically appropriate document.
- *Mean Reciprocal Rank (MRR):* One over the rank of the first suggestion where the user liked both the description and the geographically appropriate document.
- *Time-Biased Gain (TBG):* This measure provides a unifying framework for information retrieval evaluation, generalizing many traditional effectiveness measures while accommodating aspects of user behavior not captured by these measures. By using time as a basis for calibration against actual user data, time-biased gain can reflect aspects of the search process that directly impact user experience, including document length, near-duplicate documents, and summaries [2].

According to Table 2, DuTH_B yielded better results than DuTH_A in all the evaluation measures. In any case, our two runs seem very promising considering the Best, Median and Worst results of the 34 submitted runs, provided for all three measures. The comparison of our results with the Best and Median results are shown in Table 3. In P@5, DuTH_B performed equal or better then the Median in 209 of the 223 judged context-profile pairs and achieved 47 times the best run. In MRR, DuTH_B scored equal or better than the Median in 206 of the 223 judged context-profile pairs and achieved 114 times the best run. In all the aforementioned counts, we have included the judged context-profile pairs with zero best score (11 for P@5 and MRR, and 8 for TBG).

Table 3: Number of context-profile pairs with Median-or-better and Best scores per measure.

Runs	Median-or-better			Best		
	P@5	MRR	TBG	P@5	MRR	TBG
DuTH_A	189	175	151	25	86	22
DuTH_B	**209**	206	185	47	**114**	40
Total: 223 judged context-profile pairs						

5 Conclusions

We presented a context processing method that we used in order to collect over 22,000 places for the TREC 2013 Contextual Suggestion Track using three popular place search engines: Google Places, Foursquare and Yelp. Furthermore, we proposed two methods for personalized suggestion of places/attractions with respect to given user preferences. The first suggestion model is based on a k-NN algorithm by using tf-idf weights in the calculations of places' scores (run DuTH_B). In the second suggestion model, based on Rocchio algorithm, we proposed the generation of a weighted personal query for each user that was created by using terms from the example places and the preferences of user. Then, this personal query is used to retrieve from a context the places that are suggested to the user (run DuTH_B).

In the TREC evaluation results, both approaches seem very promising. DuTH_B (i.e. the Rocchio-like approach) performed better than DuTH_A. Compared to other groups, DuTH_B scored almost firmly above the median (in P@5 and MRR) and achieved the best results in almost half of the judged context-profile pairs (at MRR). In the final TREC system rankings, we are the 2nd best group in MRR and TBG, and 3rd best group in P@5, out of 15 groups in the category we participated. As first-time participants, we are very satisfied with these results.

Acknowledgements. The present work was partially funded by the project ATLAS (Advanced Tourism Planning), GSRT/CO-OPERATION/11SYN-10-1730.

References

[1] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[2] A. Dean-Hall, C. L. A. Clarke, P. Thomas, and J. Kamps. Evaluating contextual suggestion. In *Proceedings of the 5th International Workshop on Evaluating Information Access (EVIA)*, 2013. http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings10/pdf/EVIA/09-EVIA2013-DeanHallA.pdf.

[3] O. Dunn and V. Clark. *Applied statistics: Analysis of variance and regression*. Wiley, 1974.

[4] G. Hubert and G. Cabanac. Irit at trec 2012 contextual suggestion track. In E. M. Voorhees and L. P. Buckland, editors, *Text REtrieval Conference (TREC' 12)*, Gaithersburg, USA, Nov. 2012. National Institute of Standards and Technology (NIST).

[5] R. Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '93)*, pages 191–202, New York, NY, USA, 1993. ACM.

[6] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[7] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System – Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, Englewood Cliffs, NJ, 1971.

[8] M. Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):111–147, 1974.

[9] TREC Contextual Suggestion. Trec 2013 contextual suggestion track guidelines, October 2013. `https://sites.google.com/site/treccontext/trec-2013-guidelines`.

[10] W. E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods*, pages 354–359. American Statistical Association, 1990.