Suggesting Points-of-Interest via Content-Based, Collaborative, and Hybrid Fusion Methods in Mobile Devices

Article *in* ACM Transactions on Information Systems · September 2017

DOI: 10.1145/3125620

CITATIONS
17

READS
77

2 authors:

Avi Arampatzis
Democritus University of Thrace
108 PUBLICATIONS **1,513** CITATIONS

SEE PROFILE

Georgios Kalamatianos
Uppsala University
9 PUBLICATIONS **44** CITATIONS

SEE PROFILE

Suggesting Points-of-Interest via Content-Based, Collaborative, and Hybrid Fusion Methods in Mobile Devices

AVI ARAMPATZIS and GEORGIOS KALAMATIANOS, Democritus University of Thrace

Recommending venues or points-of-interest (POIs) is a hot topic in recent years, especially for tourism applications and mobile users. We propose and evaluate several suggestion methods, taking an effectiveness, feasibility, efficiency, and privacy perspective. The task is addressed by two content-based methods (a Weighted kNN classifier and a Rated Rocchio personalized query), Collaborative Filtering methods, as well as several (rank-based or rating-based) methods of merging results of different systems. Effectiveness is evaluated on two standard benchmark datasets, provided and used by TREC's Contextual Suggestion Tracks in 2015 and 2016. First, we enrich these datasets with more information on venues, collected from web services like Foursquare and Yelp; we make this extra data available for future experimentation. Then, we find that the content-based methods provide state-of-the-art effectiveness, the collaborative filtering variants mostly suffer from data sparsity problems in the current datasets, and the merging methods further improve results by mainly promoting the first relevant suggestion. Concerning mobile feasibility, efficiency, and user privacy, the content-based methods, especially Rated Rocchio, are the best. Collaborative filtering has the worst efficiency and privacy leaks. Our findings can be very useful for developing effective and efficient operational systems, respecting user privacy. Last, our experiments indicate that better benchmark datasets would be welcome, and the use of additional evaluation measures—more sensitive in recall—is recommended.

CCS-Concepts: • **Information systems** → **Recommender systems**; *Location based services*; *Mobile information processing systems*; *Collaborative filtering*; *Nearest-neighbor search*; *Rank aggregation*; Retrieval models and ranking; • **Security and privacy** → *Privacy protections*;

Additional Key Words and Phrases: Contextual suggestion, recommender systems, privacy

ACM Reference format:
Avi Arampatzis and Georgios Kalamatianos. 2017. Suggesting Points-of-Interest via Content-Based, Collaborative, and Hybrid Fusion Methods in Mobile Devices. *ACM Trans. Inf. Syst.* 36, 3, Article 23 (September 2017), 28 pages.
https://doi.org/10.1145/3125620

1 INTRODUCTION

Smart devices, and in particular smartphones, are ubiquitous devices, more than just communication tools, as they combine the features of a cell phone along with computing functionalities.

Authors' addresses: A. Arampatzis and G. Kalamatianos, Database & Information Retrieval unit, Electrical & Computer Engineering Department, Democritus University of Thrace, University Campus at Kimmeria, 67 100 Xanthi, Greece; emails: avi@ee.duth.gr, georkalamatianos@gmail.com.

ACM 1046-8188/2017/09-ART23 $15.00
https://doi.org/10.1145/3125620

Ubiquitous devices mingle, and sometimes interfere with a person's everyday life. Smartphones hold on to a plethora of personal, yet heterogeneous data generated from various software applications, hardware components, and embedded sensors, constituting a very rich set of potentially private information. Consequently, a smartphone can be considered as a well-informed—about its owner's interests—ubiquitous digital companion that comprises a diverse set of sensors and strong computational and networking capabilities. Thanks to these features, information available to a smartphone can be used in many circumstances to assist its owner with suggestions and recommendations.

In this work, we focus on the sets of personal data of smart devices and consider how it can be used to make touristic contextual suggestions to their owners. Whatever may attract a tourist can be considered as a Point-Of-Interest (POI), and tourists often are overwhelmed by the large number and variety of POIs in the places they visit. Typically, a tourist has to make her choices in short time and without being fully informed about her options regarding the available POIs. Assuming that a contemporary tourist owns a smartphone equipped with a number of sensors, and is familiar with mobile applications (apps), mobile recommender systems can become valuable tools. In fact, mobile tourism services is currently a very active research topic; for a collection of related papers, see, e.g., [59].

A recommender system for tourism aims to advise its user on which POIs to visit while staying in a certain area. Such suggestions are produced based on inputs, and a simple approach would be to ask the user about her preferences, and include information such as user needs, interests, and constraints, enter those into some system, and then have the system correlate these preferences with POIs which have similar parameters [30]. Another option is to use additional input evaluations and ratings of other tourists with similar interests in a collaborative filtering fashion [51]. Additionally, travelers who are in close spatial and temporal proximity often share common travel interests or needs [21, 61]. A common requirement of all these approaches is that users have to enter their personal information, POI ratings, and build their profile into some system.

Another, possibly more important, requirement of existing recommender systems is that user profiles have to be stored and managed by the recommender service. User profiles, however, contain personal data, some of which may be considered *sensitive personal data* according to the Data Protection Directive 95/46/EC [29], the current draft of the forthcoming General Data Protection Regulation (GDPR),[1] which regulates the processing of personal data within the European Union, and similar regulations which hold in other regions. This, in turn, raises privacy concerns to potential users of such systems.

Moreover, the privacy issues can be further aggravated when user profiles are combined among recommender systems in order to expand the data pool and enable more intelligent recommendations [64]. The privacy concerns are even more serious in mobile recommender systems, due to the wide range of sensitive personal data (e.g., location) that are available to mobile platforms and can be accessed by mobile apps and transparently uploaded to remote servers. In fact, user awareness of threats against location and identity privacy aspects have been recognized as one of the greatest barriers to the adoption of context-aware services [10].

The exact challenge addressed in this work is to propose and experimentally compare POI recommendation/suggestion methods for tourism, paying attention also to privacy. The targeted functionality is mostly in line with the Contextual Suggestion Track of the Text Retrieval Conference (TREC) (see, e.g., [22, 24, 32]). In other words, we focus on a single but common situation [24]:

[1] http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf (accessed on 20 April 2016).

> 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?*

Earlier TRECs (2012–2014) considered as context mainly the user's location (some temporal information was coming in and out across the years) and asked for suggestions taking into account also the user interests via personal past preferences. In TREC 2015 and 2016, the track organizers put more effort into providing richer contextual information in requests including also trip type, duration, travel companions, and season.

TREC's experimental setup considers each user in isolation, thus a recommender system could run entirely in her mobile device, given all the POI data of a general area of interest. Such a setup has some desirable privacy properties. In this work, we extend the setup to also consider POI ratings of other users in a collaborative filtering fashion. Due to the large number of POIs and users globally available, such systems usually run in a centralized service, sacrificing privacy.

We approach the problem in three ways: (a) content-based filtering, (b) collaborative filtering, and (c) hybrid methods fusing the former two. Our main contributions are the following.

- We propose a content-based filtering approach derived from Rocchio's relevance feedback method [49]. We *modified* Rocchio's method so as to suit multi-graded rating environments, which we call *Rated* Rocchio. In the environment considered, users usually rate POIs employing grading systems beyond the binary relevant/non-relevant of traditional retrieval, e.g., 5-graded: strongly interested, interested, neither interested or uninterested, uninterested, strongly uninterested. To our knowledge, no one has employed Rocchio before in our way; we first introduced this modification in TREC [27, 34].
- We compare the Rocchio method to the effectiveness of another content-based classifier, namely, a k Nearest Neighbors (kNN) classifier [4], classifying each candidate POI to one of the five grades. Here, we employ a *Weighted* kNN version, trying several weighting methods for the semantically nearest neighbor POIs. We are not aware of any other research employing kNN in this context with weights in our fashion; we first introduced it for the task in [27] in parallel with [9] in the same year's TREC (2013).
- As a collaborative filtering approach, we employ the Pearson Correlation Coefficient method, looking for similar users according to their rated POIs. While this is a standard and widely used approach in recommender systems [46], it has not been used like that in the datasets we employ, since TREC's experimental setup has discouraged collaborative approaches. So, it can serve as a baseline for future research, as well as an interesting comparison to the above-mentioned two content-based methods.
- For fusing the content-based and collaborative filtering methods, we mainly resort to the preferential voting methods of Borda Count [12] and a Condorcet system [19]. Since we have two content-based and one collaborative filtering method, we try a three-way fuse, as well as all two-way combinations. Although there exists previous research on such hybrid methods, we are not aware of any past research applying these on the datasets we employ.
- Beyond the above rank-based fusion methods, we also investigate whether rating-based fusion is feasible and effective. Note that not all of the above methods predict ratings, e.g., the Rated Rocchio produces retrieval scores. In this respect, we experimentally investigate techniques for mapping these scores to ratings, so as ratings instead of ranks can be fused across systems by, e.g., taking their average.
- All the above methods are experimentally tested on TREC-provided free datasets. However, we enrich these datasets with POI information crawled from free online services such as

Foursquare[2] and Yelp.[3] We make these extra data available, which may constitute a valuable additional resource for future research.

— For all proposed methods, we investigate the feasibility of their local implementation at the user-side (user's device), even in parts, and discuss whether user privacy can be preserved. Furthermore, considering that user devices may have insufficient computational capabilities for the task, we provide some basic theoretical complexity analysis of the methods. The analysis and discussion provided may be valuable in developing operational systems which are efficient, effective, and privacy-preserving, three highly valued features by users.

While, as we show experimentally, our proposed methods are at the state-of-the-art effectiveness-wise, we also discuss additional worse-performing setups which may be proven valuable for future research by pointing out dead-ends or possible ground-breaking ways. Furthermore, privacy-wise, the content-based methods provide very good characteristics and can be efficiently implemented in their biggest part at the user-side.

The rest of this article is organized as follows. In Section 2, we discuss past related work with respect to our approach. Section 3 presents our proposed methods for POI suggestion, followed by their experimental evaluation in Section 4. In Section 5, we summarize and discuss our findings. We conclude in Section 6, including possible directions for future research.

2 RELATED WORK

With respect to our work, we discuss past literature in mainly three domains: first, mobile recommender systems, especially ones considering user-privacy; second, approaches using extended contexts such as weather or health conditions; and third, POI suggestion in somewhat limited contextual information like in earlier TRECs.

Many previous works on context-aware venue recommendation (e.g., [43, 62, 63]) use check-in data from location-based social networks (e.g., Foursquare) to evaluate the accuracy of their recommendations, by assuming that users implicitly like the venues they visited. A more reliable and reusable test collection methodology to evaluate such systems has been developed by the TREC Contextual Suggestion track since 2012. Consequently, while POI recommendation is a well-studied topic, a significant part of the references given below is dominated by works using TREC data even outside the Track.

2.1 Mobile Recommender Systems

There is rich literature on recommendation systems and mobile recommendation systems (see, e.g., [11, 48] and the references therein for general recommendation systems). The application of mobile recommender systems in the tourism domain has also attracted a lot of interest from the scientific community (see, e.g., [13, 31] for up-to-date surveys of the field and related topics). More specifically, in [13] the authors focus on systems available for tourism applications, analyze different user interfaces, and discuss various functionalities offered by these systems.

An interesting application for tourists is the myVisitPlanner^{GR} system [47], which recommends activities to visitors along with appropriate schedules to carry out these activities during a visit. The protection of privacy is also discussed, and even though users have to disclose some personal data to the system, the amount of personal data disclosed is kept at a low level, measures are taken to avoid accidental data leaks, and the privacy policy of the application is clearly documented. Another recent work is the iGuide system [53] which aims at enabling a socially enriched

[2] www.foursquare.com.
[3] www.yelp.com.

mobile tourist guide service where its recommendations are enriched with multimedia content. In this direction, in [44] the authors propose a novel approach for multimedia recommendation and annotation.

Others have employed *privacy-by-design* approaches (e.g., [28]) to address the privacy issues of recommendation systems. Privacy by design generally refers to a holistic approach for handling the privacy issues within each step of the design, implementation, and operation of a system (see, e.g., [20], and the references therein).

2.2 Extended Context Approaches

Most of the aforementioned approaches mainly use as context the user's location, her personal preferences, or social network sources. Others, like the ones below, assume extended contexts.

An interesting application is FitYou [58], a system that makes contextual suggestions driven by healthy lifestyle choices according to the user's context and health condition. It employs a state-of-the-art matrix factorization approach and Singular Value Decomposition (SVD) over a user-category matrix. STS [14] is another context-aware system which exploits weather conditions data to generate higher quality live recommendations using again a matrix factorization approach.

In [37], the authors examine two approaches that model user preferences and context features (beyond just location) in terms of a word embedding vector space. They perform a feature ablation test, by recording the effectiveness when a single feature is removed. They find that positive venue ratings is the most important feature, whose absence reduces P@5 by 2.2%. The absence of each context feature—such as trip duration (day, night, weekend, longer), season (summer, winter, etc.), group (alone, friends, family, other), and trip type (business, holiday, other)—reduces P@5 by 1.4–1.9%. These results suggest that such context features are effective, but represent factors that are less important for users than their past venue preferences. Lastly, ablation of the gender feature only reduces effectiveness by 0.5%.

In [33], the authors study the impact of using context features on the overall performance of contextual suggestion based on parsimonious group profiling. The contextual customization is found almost as effective as past user preferences which shows that such features are quite promising in dealing with cold-start problems in the contextual suggestion task. They observe that trip duration is the most important context and the least important one is the trip type. A linear combination of the customized ranking (using extended context features) and the personalized one (using user-rated venues) is performing better than their best personalization approach in all the common IR metrics being used in their paper. Nevertheless, past user ratings vs. context seem to be best weighted (for P@5) in a 70–30% fashion, respectively, showing that ratings are still more important (more than twice) than context.

In our work, our considered context is more in line with the earlier TRECs, i.e., it cannot be considered as extended: we only consider the user's location. In any case, according to the works mentioned above, the past user preferences seem to be much more important than extended context information. Given the specific format of the challenge, we next refer to the efforts of authors who investigated this in the TREC conferences.

2.3 TREC's Contextual Suggestion Track

The Track has run for 5 years from 2012 to 2016. During the early 3 years, up to 2014, the task was evaluated on the ever-changing open-web, making experiments non-reproducible and meaningful comparisons difficult. We mostly review the literature from 2015 on, since the Track has moved to controlled/fix collections.

Researchers have usually employed different combinations of web-services such as Foursquare, Yelp, TripAdvisor,[4] and Google Places,[5] to collect data and enrich their dataset with keywords, descriptions, and useful phrases [3, 17, 38, 41, 52, 57, 60]. We also enrich the standard datasets in a similar way; moreover, we make these extra data publicly available for future research.

Aliannejadi et al. [3] build both a positive and a negative user model via SVM and Naive Bayes classifiers, using comments that were retrieved from the aforementioned web-services. They also employ a normalized count of category-tags occurring in rated POIs. Finally, they calculate a weighted average of the similarity values produced by the user profiles and the category-based profiles. McCreadie et al. [38] extract 49 attributes from data retrieved from Foursquare and use Learning to Rank methods to build profiles. They also use information retrieved from Foursquare profiles, such as usual times and seasons of visits, and textual information to suggest venues suitable for the users' time of visit and type of trip, using Factorization Machines. Yang and Fang [60] build a positive and a negative profile for each user and each POI according to comments retrieved from web-services. They attempt either using the text as a whole or a nouns-only approach. For each user-POI pair, they calculate a similarity value from a linear combination of all profile pairs. Finally, they employ manual criteria to boost or avoid POIs that belong to categories that match their contextual needs or not, accordingly.

Trees et al. [52] calculate heuristic normalized ratings for each occurring category-tag according to the user preferences, and employ them as user-profile attributes. They also employ a penalty function for frequently occurring tags to enhance variability. Chakraborty et al. [17] use the tag co-occurrence count between POIs and users as a similarity value. Wang et al. [57] create a user preferences profile and POI profiles applying different weights to terms retrieved from keywords, text, or title, while applying negative weights to terms with a negative contribution. They then rank the list of suggested POIs according to the confidence value for each returned POI. Chen et al. [18] try combined SVM and Matrix Factorization methods to classify candidate POIs as interesting to the user. Mo et al. [41] build both a positive and a negative user profile according to retrieved data for the user preferences. They then classify each POI to one of the profiles using an SVM classifier.

In summary, learning/classification methods employed by participants in TREC, vary from SVM, Naive Bayes, Learning to Rank, Factorization Machines, and others. No one seems to use *k*NN[6] or Rocchio, as we propose and evaluate in this work. Some seem to build separate positive and negative profiles—we build single ones incorporating negative information/feedback, e.g., via Rocchio; *k*NN is a "lazy" classifier, it does not build profiles.

3 SUGGESTING POINTS-OF-INTEREST

In this section, we elaborate on the methods we propose. First, in Sections 3.1 and 3.2, we revisit and revise the methods we first used in DUTH's[7] TREC 2013 participation [27], which were also used in [26]. Then, we describe standard collaborative filtering approaches for the task, in Section 3.3. In Section 3.4, we attempt to combine the results of all the aforementioned methods employing preferential voting systems.

3.1 Suggestion Model Based on a Weighted *k*NN Classifier

Motivated by the *k*NN classification method, we attempt to predict a user rating for POIs of potential interest to the user based on the actual user ratings of the k semantically nearest

[4] www.tripadvisor.com.
[5] developers.google.com/places.
[6] Besides [9], who introduced it in parallel with us [27] in the same year's TREC (2013).
[7] Democritus University of Thrace, Greece.

neighbors/POIs that the user has visited and rated before. We then rank the results based on the predicted rating. Specifically, we proceed as follows:

(1) *Indexing Rated POIs*: For each user, we generate an index containing any useful textual data collected about POIs they have rated.
(2) *Query Generation per Candidate POI*: We generate a query for each candidate POI which consists of a bag-of-words containing all useful textual information collected about each POI.
(3) *Rating Context/Candidate POIs*: For each candidate POI, we submit its generated query to the user index we built in step (1), which returns a ranked list of POIs rated by the user, scored for their similarity to the candidate POI. To compute a score p for the candidate, we use a weighted average of the nearest neighbor ratings:

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

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, and r_i is the user's rating for the i-th neighbor.

Thus, our main contribution here is the use a *Weighted* kNN method. We have tried several other weighting methods before committing to the use of the tf.idf score, as we will later see in Section 4.5.1. There, we will also motivate the value of k we choose for our current experimental setup.

Let us do some basic complexity analysis of the above method. Per user, the method requires an index of her rated POIs which can be built once and whenever new ratings become available it can be updated incrementally. kNN is a "lazy" classifier, in the sense that it does not build profiles. Suggesting POIs with kNN requires running a query per candidate POI to the index. Assuming that the user has rated only N_r of the N total/global number of POIs, the index contains N_r POIs; such an index can be built in linear time to the number of indexed items [36], i.e., $O(N_r)$. Searching the index is also linear to the number of indexed items, thus looking for the nearest neighbors of a candidate POI takes $O(N_r)$ time. If N_a is the number of available POIs in the area of interest, searching for all these POIs takes $O(N_a N_r)$.

Concerning the user-privacy that can be achieved by the method, the set of a user's rated POIs is relatively small so it can be kept locally in her device. Current mobile devices are computationally more than capable of building an index for hundreds or even thousands of rated POIs ($O(N_r)$) and update this index incrementally ($O(1)$) when the user gives a new rating. Suggesting candidate POIs is also computationally light ($O(N_a N_r)$), for N_a and N_r in the ranges of hundreds to thousands and tens to hundreds, respectively. The only information leak is the extended area of interest, i.e., the area she is located at the time of the request, since up-to-date information on the nearby N_a POIs has to get downloaded by some provider.

3.2 Suggestion Model Based on a Rated Rocchio Method

In this model, we generate a personalized query per user, encapsulating her interests. The query consists of terms weighted via a custom version of Rocchio's formula usually employed for relevance feedback [49]. We assume integer ratings r in $[r_{\min}, r_{\max}]$ with a neutral value r_{neutral}. The model is summarized in the following steps.

(1) *Personal Query Generation*: Let $\vec{D_n} =< d_{n,1}, d_{n,2}, \ldots, d_{n,T}, >$ be a training example, where T is the total number of terms in the training set of all rated POIs of a user. The $d_{n,t}$ is the weight of t term in D_n. This term weight is calculated as $d_{n,t} = 1 + log(f_{n,t})$ where $f_{n,t}$ is

the frequency of appearance of the t-th term in POI n represented by D_n. We use a *tf*-only weight as the inclusion of an *idf* term did not improve our early experiments in [27]. To calculate the personalized query $\vec{Q}$ for the user, we then use the following formula:

$$\vec{Q} = \sum_{r=r_{\min}}^{r_{\max}} \left((r - r_{\text{neutral}}) \frac{1}{|R_r|} \sum_{\vec{D_n} \in R_r} \vec{D_n} \right), \tag{2}$$

where R_r is the set of vectors of the user's preferences rated r by the user. In this way, neutrally rated POIs do not contribute to the personalized query (the coefficient is zeroed out), and positive- and negative-rated POIs contribute accordingly.

(2) *Indexing Context POIs*: For a certain area of interest, we build an index of its POIs, representing each POI by its corresponding collected textual data.

(3) *Suggesting Candidate POIs*: To suggest candidate POIs for a user within an area of interest, we submit the query generated in step (1) to the index of step (2). We suggest POIs with their descending order in the ranked list.

Thus, our main contribution here is a *generalized* Rocchio formula which can be used in multi-graded environments, which we will call, *Rated* Rocchio.

We have so far described how a Rated Rocchio personalized query can be constructed, however, there are more decisions to be made on how exactly this query may be used in order to score POIs. For example, whether one should use all query terms or only some top ones and how many, whether to retain terms with negative weights (if any), whether to use term weights at all, and so forth. We consider some of these choices as implementation details and some others as tightly connected with the data at hand, so we will present them in the content of our experimental setup in Section 4.3 and later.

In Equation (2), the ratings r (or rather their offsets from the neutral rating, $r - r_{\text{neutral}}$) function as the traditional Rocchio parameters (i.e., the ones weighting the original query against the relevant and non-relevant training sets), here weighting the rating sets in a linear way. That is, assuming a neutral rating of 2, the 4-rated training examples are weighted as $4 - 2 = 2$, the 3-rated examples as $3 - 2 = 1$, the 1-rated examples as $1 - 2 = -1$, and so on. We remind the reader that there is no original query in this task; the "original query" is the user's location, i.e., *here I am, what can I do?* Additionally, we will also experiment with a non-linear mapping of ratings to Rocchio parameters in Section 4.5.1, but this yielded no significant effectiveness differences; consequently, we resort so far to the simpler method of the linear mapping of Equation (2).

Also 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. We will come back to this when we try to fuse several suggestion models in Sections 3.4.3 and 4.5.3.

Let us now do some basic complexity analysis of the above method. Building a Rocchio profile is linear to the number of user-rated POIs, i.e., $O(N_r)$, where N_r is the number of user-rated POIs, as in the previous section. Once calculated, a user's profile can be retained and it changes only when the user gives a new or changes a rating. Note that Rocchio queries can be updated incrementally (see, e.g., [5], in $O(1)$). Suggesting POIs requires running the personal Rocchio query to the index of all N_a POIs in the area of interest, thus it takes $O(N_a)$ (building and searching an index is linear to collection size). Consequently, assuming a pre-generated user query, it takes $O(N_a)$ to generate suggestions.

Concerning the user-privacy that can be achieved by the method, the set of a user's rated POIs is relatively small so it can be kept locally in her device. Current mobile devices are computationally more than capable of building a personalized query from hundreds or even thousands of rated

POIs ($O(N_r)$), and update this query incrementally ($O(1)$) when the user gives a new or updates a rating. Suggesting candidate POIs is also computationally light ($O(N_a)$). The only information leak is again (as in the Weighted kNN method case) the area of interest, since up-to-date information on the nearby N_a POIs has to get downloaded by some provider.

Note that the index of all N_a POIs in the area of interest can be delivered ready by some provider, saving the computational load at the user's mobile device.

3.3 Collaborative Filtering

We construct an $M \times N$ Utility Matrix U where M is the number of users and N is the number of globally available POIs. Each component $u_{m,n}$ of U is the rating of user m for POI n. Per user, we calculate the average of her known ratings, and subtract this average from all her ratings. Then we fill the missing ratings as zero, assuming that the user would have given an average rating to the POIs she has not seen or rated.

For all user pairs x, y, we calculate Pearson's Correlation Coefficient ρ as a similarity measure. We then calculate each missing rating $u_{x,j}$ of user x as

$$u_{x,j} = \frac{\sum_{y,\, u_{y,j} \neq 0} \rho(x,y) u_{y,j}}{\sum_{y,\, u_{y,j} \neq 0} \rho(x,y)}. \tag{3}$$

Finally, for each request of user x, we first filter out POIs outside the location of interest, and then rank and suggest the remaining j's in their descending $u_{x,j}$ rating. The average user rating of user x can be added again, so only POIs with above-neutral rating are suggested to the user. This is a standard and widely used method for collaborative filtering (see, e.g., [46]). Note that when user ratings are re-centered around each user's average rating (like we do with the offset above), then ρ is the same as cosine similarity.

The method described above considers user–user similarities in order to predict missing ratings. The task can be also performed by considering POI–POI similarities, in a similar manner; we do not give the formula, it is very similar to Equation (3) and easy to derive.

Let us do some basic complexity analysis of the above method for a single user request. We assume some sparse matrix representation of U where zeros are not stored or contribute into the calculations below. Calculating and offsetting the user average rating takes $O(N_r)$; this can be done in advance and retained until the user gives a new or updates a rating. Updating an average can be done incrementally in $O(1)$, but offsetting again takes $O(N_r)$. For a pair of users, calculating a ρ is linear to the largest number of rated POIs of any of the two users; without loss of generality, we can assume that this takes $O(N_r)$. Thus, for the user paired with everybody else, all ρ's take $O(MN_r)$. These ρ's can be pre-calculated and retained, but practically not for very long, if the most up-to-date information is required for suggestions. The addition or modification of a rating by *any* other user would trigger a re-calculation of a single ρ in $O(N_r)$, but this would happen very often; roughly M times more often than in the content-based approaches. Thus, without loss of generality, and for comparison purposes with the content-based approaches, we will consider the update complexity here to be $O(MN_r)$. The addition or modification of a rating by the user would not happen so often, but it will take $O(N_r)$ for re-calculating and offsetting the user average rating, and $O(MN_r)$ for re-calculating *all* ρ's; thus, $O(MN_r)$.

Assuming that each user has rated N_r POIs randomly, a specific POI is rated by a user with probability N_r/N, and the expected number of ratings per POI from all users is MN_r/N. Thus, the nominator, as well as the denominator of Equation (3), each take $O(MN_r/N)$, since they are sums across all users' ratings for a POI. So, for all missing user ratings $u_{x,j}$ in the area of interest, in total we have $O(N_a MN_r/N)$ operations. Summarizing, assuming pre-calculated ρ's, suggesting POIs

on-the-fly for a single user request with collaborative filtering, takes $O(N_a M N_r/N)$; nevertheless, updates are triggered more often than the content-based methods and are more costly ($O(MN_r)$).

Concerning the user-privacy of the method, it has the worst characteristics by nature, since it requires POI ratings to be shared with other users and/or some provider. Furthermore, assuming millions or billions of potential users and millions of global POIs, the time complexity of suggesting, and especially of updating, may make an implementation of such a method at the user-side prohibited. Thus, suggestions should be made by a centralized service.

3.4 Hybrid Fusion Methods

We investigate the use of data fusion methods as a means to combine the recommendations produced by separate suggestion systems. We first resort to two voting systems, Borda Count and Condorcet, suitable for rank-based fusion, since the three suggestion systems we described above do not all produce ratings, e.g., the Rated Rocchio method produces scores. Then, we attempt to map Rated Rocchio scores to ratings so as to employ rating-based fusion techniques such as CombSUM.

3.4.1 The Borda Count Election Method. The *Borda Count* election method [19] originates from social theory in voting and has been previously used as a fusion method of separate retrieval systems [42]. For this purpose, we use the lists of suggestions as ballots and each system as a voter.

Specifically, for each suggestion of rank r we assign a score $p = n - r$ where n is the number of candidates. Although other formulas such as $p = n - (r - 1)$ and $p = 1/r$ are frequently used, $p = n - r$ provided the best results in preliminary experiments. We then add the p scores of each candidate for all ballots and re-sort the list of suggestions based on the final score. We can summarize the function of the Borda Count method in the following formula:

$$s_i = \sum_{j=1}^{m} (n - r_{i,j}), \tag{4}$$

where s_i is the final score of candidate i, m is the number of combined voters, and $r_{i,j}$ is the rank of candidate i according to the j voter/method.

3.4.2 Condorcet. A voting system is called Condorcet if it is designed to elect a Condorcet winner, which is the candidate who would theoretically win in a one-on-one encounter with any of the other candidates [19]. We employ a Condorcet system in which, for every suggested POI, we count the number of its wins over other suggested POIs, that is, the number of POIs that are ranked lower at any given system. We also count its losses, accordingly. Finally, we sort all candidates first, by descending wins, and then by ascending losses.

3.4.3 Fusing Ratings. The Weighted kNN system produces ratings. The Collaborative Filtering system produces ratings but not in the same initial scale, due to offsetting the initial ratings by the average user rating. The Rated Rocchio formula ranks POIs in some decreasing score (not rating); specifically, Language Model scores in our experimental setup, which are in $(-\infty, 0]$ in the implementation we use. Hence, it is evident that it is not always a straightforward task to combine these systems in a way that produces ratings.

The user ratings produced by the Collaborative Filtering method of Section 3.3 can be brought back to the initial scale by adding again the initial user average rating. By doing that, we are able to combine this method with the Weighted kNN method by taking the average rating predicted by the two methods for each candidate POI; this is equivalent (i.e., produces the same fused ranking) to the widely used combination method of CombSUM, which simply sums the scores of the item across rankings [8]. Incidentally, in our case, it is also equivalent to CombMNZ, since each POI occurs in all rankings. We will present the results of such an experiment in Section 4.5.3.

Concerning the Rated Rocchio scores, one could devise a method which maps the scores to ratings, e.g., by doing some form of regression on the training POI scores and ratings, and using this regression function to map scores of candidate POIs to ratings. Here, there seem to be different options for calculating the regression; we will follow an experimental approach and come back to this in Section 4.5.3.

4 EXPERIMENTAL EVALUATION

In this section, we present the experimental evaluation of our proposed methods. First, we describe the benchmark datasets we employed (Section 4.1). Then, we set out to enrich these data by collecting more information about POIs from widely known and used web services like Foursquare and Yelp (Section 4.2). In Section 4.3 we give the details of how we implemented our proposed methods given the available data and tools. In Section 4.4, we present the evaluation measures, followed by our experiments and results in Section 4.5.

4.1 Datasets

We employed the datasets provided by the TREC Contextual Suggestion Tracks 2015 and 2016. TREC provided the following data in both years.

POI Collection.

- —A complete collection of 1,235,844 POIs, each comprised of a title, ID, location ID, and URL.
- —A web-crawl of the aforementioned POI URLs, which was provided by TREC in 2016.

User Requests. Additionally, two different sets of requests were provided per year: 211 requests made by 209 users (for 2015), and 442 requests made by 238 users (for 2016). Specifically, each request is made by a specific user and contains information about the context in which it was made:

- —A request ID.
- —General location information about where the request was made (city, state, etc.).
- —A set of candidate POIs to rerank, at the specified location of interest. These were exactly 30 in 2015 requests, and variable from 79 to 119 (with an average of 96.5) in 2016 requests.
- —The intended duration of the trip (day trip, night out, weekend, longer).
- —The season in which the request was made (e.g., summer, winter).
- —The type of group that the user is traveling with (alone, friends, family, other).
- —The type of trip (business, holiday, other).

For the 2016 requests only, TREC additionally provided tags from a controlled vocabulary (e.g., bar-hopping, desert, live music) for the candidate POIs; the full set of possible tags can be found online.[8]

User Profiles. A user profile consists of the following:

- —A user ID.
- —The user's age and gender (both optional).
- —A set of venues (POI IDs) 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 the same controlled vocabulary we mentioned above. There were exactly 30 rated venues in each user profile in 2015 data, and either 30 or 60 in 2016 data.

[8] https://github.com/akdh/csttools/blob/master/tags.csv.

Table 1. Dataset and Web-Service Collection Statistics

	2015	2016
Number of users	209	238
Number of POIs per user profile	30	30 or 60
Total number of requests	211	442*
Number of requests (unique users) evaluated by TREC	211 (209)	58 (27)
Number of candidate POIs per request	30	79–119, avg. 96.5
Unique rated POIs in all user profiles	4,102	60
Unique POI candidates in all requests	5,463	18,752
Intersection of the above	771	4
Union of the above	8,794	18,808
Yelp profiles	6,929	15,295
Foursquare profiles	7,246	15,741
POIs with at least a Yelp or Foursquare profile	7,533	16,680
POIs with both Yelp and Foursquare profiles	6,642	10,100
Avg. unique/non-unique keywords per POI	11.82/14.34	17.80/22.31
Avg. unique/non-unique keywords per POI (enriched)	44.77/62.73	50.13/72.12
Enrichment percentage	+279%/+337%	+182%/+223%

Later this number was reduced to 438, since four requests were not completed by TREC assessors.

More information about these datasets can be found online.[9] Some summary statistics are given in the top part of Table 1.

4.2 Dataset Enrichment

We enrich the above TREC data by collecting more information about POIs from widely known and used web services like Foursquare and Yelp.

Firstly, we identify the POI IDs that will be useful to the experiments, namely, the set of POIs that occur in user profiles or the candidate POIs in requests. We then proceed to collect more online data about this set of POIs. The TREC-provided URLs correspond to either the official web page of the venue or its profile in a related web service (e.g., Foursquare, Yelp, TripAdvisor) For each case we proceed in a different way.

Venue Homepage URLs: For participants' convenience, TREC provided a set of web-crawls relieving us from performing this task ourselves. Consequently, instead of going into the open web, we process the provided dataset searching for the set of useful URLs that we identified earlier.

Web-service URLs: A significant percentage of the provided URLs correspond to a web-service profile, most of which are Yelp or Foursquare profiles. Some URLs point to TripAdvisor profiles 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 then attempted to match all POIs to both Yelp and Foursquare profiles. Specifically:

- —For URLs that corresponded to Yelp profiles, we used the exact coordinates provided by Yelp'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 same way to match POIs with Foursquare URLs to Yelp profiles.

[9] https://sites.google.com/site/treccontext/.

—For the rest of the URLs (i.e., other web-service URLs not corresponding to a Yelp or Foursquare profile, and venue homepage URLs), we attempted to match the POI 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 POIs that we successfully matched to any or both web-services' profiles, we retrieved and saved the complete profiles so as to have all information available to use at a later time.

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 union of POIs in all user profiles and candidate POIs in all requests), we obtain at least a Yelp or Foursquare profile—that is very good coverage. The enrichment process increases the average number of unique (non-unique) keywords per POI by 182–279% (223–337%). We will make available online these extra data to accompany and enrich the TREC data in future research.[10]

4.3 Dataset Processing, Tools, and Methods

The textual data we considered useful in order to describe each POI consist of a bag-of-words containing the metadata extracted from the venue homepage URLs (*description*, *title*, *keywords*), the Foursquare profile data (*description*, *title*, *tags*, *phrases*), and the Yelp profile data (*description*, *category*, *title*). In venue homepages, we ignored any other text present, since we have found that many homepages contain inherently uninformative content such as slogans, directions, and so forth. Also, to make things worse, a significant percentage of homepages consist of some type of multimedia, putting these venues at a disadvantage.

For indexing we used Indri version 2.2[11] with its default settings except that we enabled the Krovetz stemmer, as suggested by [26]. In the Weighted kNN method, we run candidate POI queries to the index of rated POIs using the tf.idf retrieval model; the choice of tf.idf was decided in preliminary experiments, as we will later see. In the Rated Rocchio method, we run personalized Rocchio queries to the index of candidate POIs using Indri's default Language Model retrieval; this is known to be the most effective ranking option in Indri. After we generate a Rated Rocchio query, we also use the resulting term weights when we run it. For example, using the Indri Query Language notation:

$$\#weight(1.5\ restaurant\ 1.2\ steak\ \ldots\ 0.2\ good).$$

We also apply a cut-off threshold of 20 highest weighted terms for each query to achieve a more precise query and a faster response. We retain terms with negative weights, if any; nevertheless, these are unlikely to occur in the top-20.[12] The number 20 emerged from preliminary experimentation with the 2015 data, and it is also a commonly used value for relevance feedback.

Note that the Rated Rocchio method, as described in Section 3.2 in a general case, requires an index of all POIs in the location of interest (context) which it ranks. The benchmark datasets employed require one to rank only a given subset of the context POIs (i.e., the candidate POIs). In this respect, after we submit the personalized Rated Rocchio query to the context POI index, we filter out all non-candidate POIs and suggest the rest. Similarly, in the Weighted kNN and collaborative filtering methods, we rate and suggest only the candidate POIs.

[10] http://poi.nonrelevant.net.

[11] www.lemurproject.org.

[12] In retrospect, we recommend removing terms with negative weights, just in case, and in order to be in line with the relevance feedback practice.

4.4 Evaluation Measures

We evaluate the quality of our suggestions employing the NDCG@5, P@5, and RR measures. These three measures were used in the TREC Contextual Suggestion Track 2016, while 2015's Track used only the latter two. The definitions of these measures are as follows:

— *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 (i.e., graded relevance) [36].
— *Precision at Rank 5 (P@5):* The fraction of relevant suggestions within the top-5 results.
— *Reciprocal Rank (RR):* The inverse rank of the first relevant suggestion.

These measures are macro-averaged over all requests; next, we will simply talk of NDCG@5, P@5, and MRR, for the macro-averaged versions. TREC provides the official ground truth on their site.[13,14] For 2015, they provide judgments for all 211 requests that come with the dataset, while for 2016 they officially evaluated on a subset of only 58 requests out of 442.[15] We evaluate using the same request-sets as TREC.

Regarding NDCG@5, note that while the TREC 2016 benchmark dataset is 5-point graded, the TREC 2015 dataset is binary graded. While NDCG@5 is still defined on binary judgments, it kind of loses its main feature (i.e., evaluating multi-graded relevance) in the 2015 dataset. Actually, P@5 and MRR have a similar "weakness," this time on the 2016 dataset: the 5-point graded relevance has to be thresholded at the neutral rating in order to calculate these measures.

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 RR. 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 a single POI (RR).

4.5 Experiments

In this section, we summarize the results of our experiments. First, in Section 4.5.1, we describe some preliminary runs we did in order to decide (a) what setups make sense or are interesting to compare, and (b) set some parameters. Then, the final runs are presented which are also tested for statistical significance over a baseline (Section 4.5.2). Some additional experimental investigation in fusing runs is presented in Section 4.5.3. Next, we investigate where our system stands against other participants in the TREC Contextual Suggestion Tracks (Section 4.5.4).

4.5.1 Preliminary Runs. In the Weighted-kNN method (WkNN), we used $k = 7$, since for some candidate POIs Indri returned no more than seven similar venues. That is because Indri does not rank POIs that bear no similarity (no matching keyword) to the query. For weights, we investigated the use of the plain tf.idf weight against two alternative weighting methods. First, we tried to use the Language Model (LM) for retrieval and its scores as weights. Indri returns LM scores in $(-\infty, 0]$, being log probabilities. We normalized each score s as 2^s and used these as weights, which

[13] http://trec.nist.gov/act_part/tracks/context/qrels-batch.txt (2015)—requires an active participant's password.
[14] http://trec.nist.gov/act_part/tracks/context/qrels.txt (2016)—requires an active participant's password.
[15] The ground truth of the rest of the requests was built based on a shallower pool depth, so they are not as good as the official 58 requests in terms of re-usability and pool bias [32]. Note also that of the 442 requests in 2016, 211 are these of 2015 but with extra candidates added as noise; their qrels are the same in both years. Thus, 211 + 227 (new in 2016) = 438, plus four not-completed by assessors, i.e., 442 in total. The breakdown of the new 2016 requests is 227 = 58 (officially evaluated) + 169 (with a shallow pool depth).

performed worse than using Indri's tf.idf retrieval model and its positive scores in $[0, +\infty)$. Then, we also tried another transformation, i.e., to use 2^s for each tf.idf score s—this gave similar results to plain tf.idf. Comparing the plain tf.idf weights to unweighted kNN showed that tf.idf weights improve effectiveness a bit in all three measures, nevertheless, the differences were statistically insignificant. In this respect, below we go with the plain tf.idf-weighted kNN.

In the Rated-Rocchio method (RRocchio), we investigated two different rating weights: (a) $r - 2$, where r is the rating, and (b) the mapping $\{0, 1, 2, 3, 4\} \rightarrow \{-2, -1, 2, 4, 8\}$ from ratings to weights. The former considers equal contributions of positive/negative ratings in feedback while discarding all neutral ratings of 2. The latter simulates the widely and empirically known effective setting of 4 to 1 ratio between positive and negative feedback for Rocchio, with a little twist: it boosts the neutral ratings to weigh half of "interested," targeting at increasing recall. Both methods yielded comparable performance with statistically non-significant differences—we selected the former and simpler method ($r - 2$) due to its slightly better effectiveness numbers. We truncate the Rocchio query to the top-20 terms with the largest Rocchio weights, as 20 is a commonly used value for relevance feedback which seemed to work well in a preliminary test with the 2015 data. Using all positive terms without a top-20 cutoff, reduced all measures significantly. We retained any top-20 terms with negative weights, if any; discarding those gave immaterial effectiveness differences since they occur very infrequently.

In the Collaborative Filtering (CF) method, preliminary runs using user–user similarity showed that the method fails completely in the 2016 dataset; consequently, we will not present these results. As shown in Table 1, the unique rated POIs in all user profiles are only 60—these are too few in relation to all unique candidate POIs occurring in requests (18,752), and their overlap is only 4. So, for almost all but four POIs, practically no predictions can be made. Consequently, we also cannot have any fusion results involving user–user CF on the 2016 dataset. We also tried CF by considering POI–POI similarities, but it failed in both datasets. Most of the candidate POIs ($5,463 - 771 = 4,692$ out of 5,463, i.e., 85.9% in 2015, and $18,752 - 4 = 18,748$ out of 18,752, i.e., almost 100.0% in 2016) do not have a single user rating; all those are deemed equally similar to each other with all their ratings equal per user (per utility matrix's row). In summary, we will only present results for user–user CF on the 2015 dataset.

In other preliminary runs with user–user CF, we tried one variation of the method described in Section 3.3. Instead of offsetting all ratings of a user with their average, it makes sense and is tempting to offset by r_{neutral} (=2, in our case), and use cosine similarity instead of ρ. This corresponds to assuming that the user would have given a neutral rating to POIs she has not seen or rated. Nevertheless, our preliminary experiments gave slightly worse results in 2015, i.e., NDCG@5/P@5/MRR of 0.5358/0.5289/0.6659, than the standard method we adopt below.

4.5.2 Final Runs. Table 2 summarizes our final results. The best results per measure (ignoring the Median/Best TREC runs in the last two rows) are in bold typeface.

All three individual methods (second batch of rows in the table), WkNN, RRocchio, and CF (ignore the RRocchio(MI) run for now—we will come back to it later) perform similarly with no statistical significant differences. We select the WkNN method (WkNN) as a baseline, as it seems to be the best method in two out of three measures (NDCG@5 and P@5) in 2015, and the best in NDCG@5 in 2016 (the main measure used by TREC that year), in absolute numbers (non-significantly). Note that this is a rather hard baseline not trivial to beat, considering that it is an above-median run in TREC 2015 and near the best run of TREC 2016 or better than the best in NDCG@5. The Rated-Rocchio method (RRocchio) seems to perform slightly better than WkNN in 2016 in two out of three measures, again in absolute numbers. In 2015, CF seems to perform in-between WkNN and RRocchio but is better than both in MRR.

Table 2. Results (2015 and 2016). Significance-Tested with a Bootstrap Test, One-Tailed, at Significance Levels 5% ($^{\triangle\triangledown}$), 1% ($^{\blacktriangle\triangledown}$), and 0.1% ($^{\blacktriangle\blacktriangledown}$), against the W$k$NN Baseline

	2015			2016		
	NDCG@5	P@5	MRR	NDCG@5	P@5	MRR
WkNN (non-enriched/original data)	0.5223$^{\triangledown}$	0.5109$^{\triangledown}$	0.6763^{-}	0.2576$^{\triangledown}$	0.3966$^{\triangledown}$	0.6016^{-}
RRocchio (non-enriched/original data)	0.5399^{-}	0.5251^{-}	0.7075^{-}	0.3262^{-}	0.4586^{-}	0.6734^{-}
WkNN [selected baseline]	0.5553	0.5450	0.6774	**0.3388**	0.4690	0.6697
RRocchio	0.5305^{-}	0.5175^{-}	0.6795^{-}	0.3306^{-}	0.4724^{-}	**0.6801**$^{-}$
RRocchio(MI)	0.4905$^{\blacktriangledown}$	0.4815$^{\blacktriangledown}$	0.6485^{-}	0.3360^{-}	**0.4828**$^{-}$	0.6367^{-}
CF	0.5468^{-}	0.5374^{-}	0.6814^{-}	—	—	—
BC CF–WkNN	0.5653^{-}	0.5536^{-}	0.7118^{-}	—	—	—
BC WkNN–RRocchio	0.5542^{-}	0.5299^{-}	0.7149$^{\triangle}$	0.3232^{-}	0.4552^{-}	0.6165^{-}
BC CF–RRocchio	0.5717^{-}	0.5573^{-}	0.7094^{-}	—	—	—
CombSUM CF–WkNN	0.5794^{-}	**0.5640**$^{-}$	0.7180$^{\triangle}$	—	—	—
CombSUM WkNN–RRocchio(LRonMI)	0.5048$^{\blacktriangledown}$	0.4796$^{\blacktriangledown}$	0.6812^{-}	0.3319^{-}	**0.4828**$^{-}$	0.6023^{-}
CombSUM CF–RRocchio(LRonMI)	0.5346^{-}	0.5261^{-}	0.6732^{-}	—	—	—
Three-way Borda Count (BC)	**0.5830**$^{\triangle}$	0.5630^{-}	**0.7414**$^{\triangle}$	—	—	—
Three-way Condorcet	0.5817$^{\triangle}$	0.5611^{-}	**0.7414**$^{\triangle}$	—	—	—
Three-way CombSUM	0.5397^{-}	0.5223^{-}	0.7026^{-}	—	—	—
Median TREC run	0.5356	0.5204	0.6758	0.2720	0.4120	0.5927
Best TREC run	0.6055	0.5858	0.7404	0.3306	0.5069	0.6854

The first two lines of Table 2 present the WkNN and RRocchio runs on the original data provided by TREC, i.e., before our enrichment process described in Section 4.2. It can be seen that enrichment helps WkNN significantly, especially in 2016, but unexpectedly it makes no significant differences in RRocchio. Looking at our data, on the one hand this seems to be due to the top-20 term cutoff in RRocchio queries which keeps them from changing too much when training data are enriched. On the other hand, while enriching the test data may have helped recall, the currently used evaluation measures are not very sensitive to it. Regarding WkNN, richer representations seem to help in better identifying a POI's nearest neighbors.

RRocchio is attempting to model the user's interests (likes and dislikes) using a single weighted query of ultimately 20 terms (after filtering). This seems like an impoverished representation of the user's interests across venues of different types (restaurants, attractions, shops, events, etc.). The WkNN approach seems more granular. So, for example, if the candidate POI is a restaurant, it focuses on the user's interests (likes and dislikes) only with respect to restaurants (the nearest neighbors). Likewise, if the candidate POI is a clothing store, it focuses on the user's interests (likes and dislikes) only with respect to clothing stores, and so forth. Nevertheless, in order to capitalize on this arguably extra granularity, richer representations are needed. For example, more clothing related keywords are needed in order to effectively find other clothing stores (nearest neighbours).

The Borda Count (BC) two-way combinations are not significantly different than the baseline (although they seem to perform in-between or better than the runs they combine), except WkNN–RRocchio which provides an improved MRR in the 2015 dataset. So, two-way fusion seems to help the first relevant suggestion, only in 2015. We narrow down our experimental investigation to using only BC, since Condorcet two-way fusion gave practically the same results as BC (so they are not presented for simplification) and marginally worse than BC in three-way fusion.

Both BC and Condorcet methods for fusing all three runs (three-way merge) yield significantly better effectiveness than any individual run in all measures except P@5 (although it improves in

both fusion methods); fusion seems again to promote especially the first relevant suggestion (i.e., more significant improvements in MRR).

4.5.3 More Fusion Runs. The only obstacle in using fusion based on, e.g., simple averaging of ratings across all methods, CombSUM, CombMNZ, and so forth, is that the RRocchio method produces scores not ratings. One could devise a method which maps the scores to ratings, e.g., by doing some form of regression on the training POI scores and ratings, and using this regression function to map scores of candidate POIs back to ratings. Here, there seem to be different options for calculating the regression.

In a first attempt, we run the RRocchio query to an index containing all and only the training POIs in order to get the score/rating data-points for the regression. Then we run the RRocchio query to the index containing only the candidate POIs and use the previously estimated regression function to map scores to ratings. We have tested this using linear regression, however, it fails completely. The problem seems to be that the score range of the training POIs produced by the former index is completely different than the one produced by the latter index. For example, the former index produces scores in $(-5.8, -5, 2)$ while the latter index, for a request, produces scores in $(-4.9, -4.1)$. In this case, the regression line estimated in the former index predicts ratings >4 for all candidate POIs in the latter index. This happens for most requests; the score range of the candidate POIs is either above or below the score range where the regression line is estimated, producing out-of-range ratings for all candidate POIs; in some cases so far out as -10 or $+30$. Even when we truncate these into $[0, 4]$, all candidate POIs fall at the same side, i.e., they are all rated as either strongly interesting or strongly uninteresting. Proceeding like that to fusion also fails completely.

In a second attempt to overcome the problem of the first attempt, we tried to use a *single* index for calculating and using the linear regression function. We built an index containing the union of training and candidate POIs, per request. Now, while running a Rocchio query to this index produces compatible scores—or in the same range—for both training and candidate POIs, two other problems occur. First, the estimated linear regression lines for a few requests have negative slopes, reversing the corresponding rankings. Mixing non-training POIs into the index of training ones seems in some cases to "spread" the training ones into the ranking in a way that higher rated ones occur below lower rated ones, producing the negative slope. This seems to be an effect of the changed background statistics (document frequencies). In the few cases that this happens, we "fix" it by assigning to all candidate POIs the rating corresponding to the candidate POI with the least rating, preserving like that the ranking.

Second, the sub-ranking of candidate POIs (ignoring the training POIs) produced by this "beefed-up" index is different than the one produced by the index containing only the candidate POIs, since the background statistics (document frequencies) change by the addition of the training POIs to the index. While this may not necessarily be a problem in a first glance, the quality of this sub-ranking is inconsistent in our experimental setup: our control run, RRocchio(MI)[16] in Table 2, gave NDCG@5/P@5/MRR of only 0.4905/0.4815/0.6485 in 2015, i.e., significantly worse than the straightforward RRocchio, but 0.3360/0.4828/0.6367 in the 2016 dataset, which is a solid run achieving even the best P@5 (with an insignificant loss of MRR).

All in all, the second attempt seems more promising, although it varies in consistency. From both attempts, we can conclude that the distribution of document frequencies in the candidate set can be quite different than the training set. The matter deserves further investigation in the future.

[16]MI denotes Mixed Index, and LRonMI (later) denotes Logistic Regression on Mixed Index.

We will proceed to CombSUM fusion using the second method for normalizing RRocchio scores to ratings, i.e., linear regression on a mixed index, RRocchio(LRonMI).

Concerning the two-way CombSUM runs, the inferior quality ranking of RRocchio(MI) in 2015 influences more negatively its combination with W*k*NN, which is significantly worse than the baseline in NDCG@5 and P@5 but holds up in MRR. The same combination in 2016 is not significantly different than the baseline, although it achieves the best P@5 of all runs. The CombSUM CF–W*k*NN is a well-performing run with significantly better MRR than the baseline, and the best P@5 of all runs in 2015. Comparing the BC to CombSUM two-way combinations, we can conclude that they are competitive, leaving aside the inconsistency of the RRocchio(LRonMI) method. Nevertheless, three-way CombSUM is left behind by BC and Condorcet.

In summary, while fusing predicted ratings across recommenders is a promising method with more potential than voting methods, mapping scores to ratings where this is needed is not trivial; there is certainly room for improvement here. When score mapping is not needed, e.g., in CF and W*k*NN, rating-based fusion seems superior. For example, the combination of CF with W*k*NN performs better when a rating-based CombSUM fusion is used (CombSUM CF–W*k*NN) rather than a rank-based Borda Count fusion (BC CF–W*k*NN).

4.5.4 TREC Runs. TREC has run a setup similar to the one we consider in this article. TREC 2015 called this "batch experiment," while TREC 2016 called it "Phase 2 experiment." During batch or Phase 2, participants were presented with the task of re-ranking a set of candidate venues, for each request provided.

Table 2 compares the effectiveness of our system to the systems of other participants in the TREC 2015 and 2016 Contextual Suggestion Tracks. One, however, should be careful in drawing conclusions about our proposed suggestion methods from this comparison due to other participating groups using different pre-processing and/or data enhancement of the POI collection. This comparison rather serves as an indication of where our system stands as a whole, including data processing and enhancement.

For TREC 2015, Median/Best runs for P@5 and MRR are taken from [22], while for NDCG@5 we calculated them ourselves since the NDCG@5 was not used in TREC 2015. That year in the Batch task (i.e., the task as described in this article), 17 teams participated with a total of 30 runs; we did not participate. For TREC 2016, Median/Best runs are taken from [32]; 13 teams participated in Phase 2 experiments (i.e., the task as described in this article) with a total of 30 runs (including our team with three runs).

For 2015, our individual methods seem to generally stand above the median run, but closer to it than to the best run. The fused methods are better, generally between the median and best runs. The three-way merges are better than all the two-way ones in all measures; they even surpass the best TREC run in MRR. Again, fusion seems to help more the first relevant suggestion.

For 2016, our individual methods are generally close to the best runs. Both W*k*NN and RRocchio beat the best run in NDCG@5, which is the main measure used by TREC. Note that TREC's best NDCG@5 of 0.3306 comes from our official RRocchio run (runtag `DUTH_rocchio` in [32, Table 3]) which is exactly the same as the one presented in Table 2. We also officially submitted a run with W*k*NN (`DUTH_knn`) and a fused BC W*k*NN–RRocchio run (`DUTH_bcf`), but we later found a bug in `DUTH_knn` which hurt its effectiveness[17] so also the `DUTH_bcf` is invalid; that was unfortunate since, according to Table 2, W*k*NN achieves better NDCG@5 than `DUTH_rocchio`. In summary, despite the "buggy" `DUTH_knn` and invalid `DUTH_bcf` runs, our system officially ranked us in TREC

[17] The official "buggy" `DUTH_knn` yielded 0.3116 NDCG@5, 0.4345 P@5, and 0.6131 MRR.

Table 3. Method Time-Complexities per Subtask, and Privacy Leaks

	Time complexity			Privacy
	Pre-process	Suggestion	Update	**leak**
Weighted kNN	$O(N_r)$ index rated POIs	$O(N_a N_r)$ run all candid. POI queries	$O(1)$ incr. index update on new user rating	area of interest
Rated Rocchio	$O(N_r)$ train query	$O(N_a)$ index candidates and run query	$O(1)$ incr. query update on new user rating	area of interest
Collab. Filt.	$O(MN_r)$ calc. user's avg. rating & ρ's	$O(N_a MN_r/N)$ calc. all ρ's and weight-average ratings	$O(MN_r)$ update ρ's & avg. or ρ on *any* user's rating	area + all user's ratings

2016 as the first team in NDCG@5 (`DUTH_rocchio`), third best team in P@5 (`DUTH_rocchio` and `DUTH_bcf`), and second best team in MRR (`DUTH_rocchio`), out of 13 participating teams.

Furthermore, employing earlier versions of our proposed individual methods, we had participated in TREC Contextual Suggestion 2013 [24] where our highest achievements were: second best team in MRR and TBG[18] measures, and third best team in P@5, out of 15 teams in the participated category/task/phase. We neither present those results here nor experimented any further with that setup, since the dataset used at that time (i.e., the open web) is non-reusable for offline (i.e., after-TREC or outside-the-pool) experiments, being non-static and with "thin" relevance judgments.

In summary, the experimental evaluation in the context of all three TRECs, in two of which we participated, reveals that our approach is either the state-of-the-art or—depending on the evaluation measure considered—comparable to the best methods in the literature.

5 DISCUSSION

5.1 Effectiveness

Overall, looking at our effectiveness results, there is something worrisome: three radically different methods achieve similar effectiveness in the 2015 dataset with statistically insignificant differences. This also happens in the 2016 dataset for the two content-based methods (we remind one that CF fails here due to rating sparsity), despite the fact that the organizers tried to pool deeper. This may point to weaknesses inherent to the benchmark dataset(s) and/or limitations in the evaluation measures used. The similarities between the two content-based methods could be more easily accepted, since the methods use the same representation space (i.e., textual data) for POIs. Nevertheless, CF does not use any textual data; it only uses the users' ratings in profiles. We further investigated on why this may be happening.

From track's overview papers [22, 32] we found out that the 2015 dataset contains very high quality candidate suggestions associated with its 211 requests, and that one could get very good results by either randomly ranking the candidates or just returning them in the same order as they come in the json file provided. The 2016 dataset does not have this problem. This is also confirmed by the much better results obtained overall (by us and TREC participants) in the 2015 than in 2016 dataset; e.g., TREC's Median and Best NDCG@5 almost halves from 2015 to 2016.

[18]TREC 2013 employed a modified Time-Biased Gain (TBG) metric instead of the NDCG@5 measure; for more information on TBG, see [23, 24, 50].

Our own investigation of our CF runs also supports the above. In 2015, according to Table 1, the overlap between the rated POIs in profiles (4,102) and candidates to be ranked (5,463) is only 771 POIs. In the way Equation (3) works, it manages to predict ratings for these 771 POIs, while for the rest 4,692 ($= 5,463 - 771$), or 85.9%, it predicts the same neutral rating for all. Assuming a uniform distribution of the overlap POIs across requests, that is 3.6 POIs per request (i.e., 771 divided by 211). Consequently, what our CF runs do is to bring up in the top-5 on average 3.6 POIs with well-predicted ratings, and then fill up the remaining one or two rank positions with randomly selected neutrally deemed POIs. Since all candidate POIs are of high quality, this produces good results. The fact that all three measures are early-precision sensitive also helps to allow this. If looser cut-offs are used, e.g., NDCG@20, the CF method is expected to be less effective than the content-based methods. Nevertheless, since the current measures are more suitable for evaluating mobile users, we have not tried this.

Now, despite the very high quality candidates in 2015, another question arises: why the ceiling of effectiveness is seen at the current levels (around 60% for NDCG@5/P@5 and 75% in MRR in 2015; and 33%, 50%, and 70%, for the three measures respectively, in 2016) and not higher, e.g., closer to 100%. This question is more important for the content-based runs, since these should have been capable of discriminating most candidate POIs by using their textual representation. Here, we hypothesize that there may be category biases; for example, profile POIs belong to either museum or bar categories while candidate POIs belong to either museum, bar, or additionally, hotel, zoo, and other categories. If this is true, we can train so we can rate POIs for some categories, while we completely miss others. This would explain the current effectiveness ceiling seen, especially in 2015, although the same argument applies to the 2016 dataset. According to the Track's organizers, although profile and candidate POIs were not collected based on categories, there could be some judged categories in the candidates that were not judged in the profiles and vice versa. The diversity problem of contextual suggestion has not yet been investigated by the Track.

One could argue that the ceiling of effectiveness can be pushed further up by using more context (e.g., type, duration, season of the trip, gender and age of the user). So far we have neither used such information, nor the category tags provided by the users for the POIs in their profiles. However, we do not believe that these would have been very valuable in the current benchmark datasets with the current evaluation measures. Some of this extra information seems correlated to profile POIs, e.g., category tags, age, gender; for example, kids would have already seen and rated a zoo or two, and 20–30-yr-old males interested in bars would have rated many of them. Nevertheless, season or duration of the trip seem like useful attributes; for example, open-air zoos could be excluded from suggestions made in winter months at cold locations, or consider the user request/context *"I am here for a day, sparing a couple of hours, what is the single most important must-see?"*. In any case, we believe the early-precision sensitivity of the current evaluation measures would not allow one to measure these attributes' usefulness. Note that the MRR is already at the "diminishing returns" area of 70–75%.

5.2 Efficiency

In Section 3, for each of the individual (not fused) methods, we have given a basic computational complexity analysis, per user request, based on the numbers of rated POIs of the user (N_r) and available POIs in the area of interest (N_a), and additionally, the total/global numbers of POIs (N) and users (M) in a CF system. The task has been broken into three subtasks: pre-processing (e.g., building necessary indices, queries, calculating user–user similarities), suggesting POIs, and updating actions required when user(s) provide new or update ratings. We have assumed that calculating suggestions always uses the most up-to-date profile and context information. These complexities together with their corresponding actions are summarized in Table 3.

Content-based methods have been deemed as faster than CF in all three subtasks. Content-based methods provide pre-process times linear to N_r and incremental updates in $O(1)$, while the CF method is mainly hit by having to consider all users M in these steps. Concerning the most important subtask of suggesting which has to be performed on-the-fly (users do not like to wait), the faster method is Rated Rocchio (linear to N_a), while WkNN is further hit multiplicatively by N_r due to the nearest neighbor search; CF is further hit or improved—depending on the specific values—multiplicatively by the ratio of global users to global POIs (M/N), since it requires one to "see" everything. These time complexities are roughly confirmed by the running times of our experiments in this article.

Let us plug in some real numbers. Factual's Global Places Data[19] currently (January 2017) contain over 100 million POIs in 50 countries. The city of Los Angeles returns around 205,000 POIs in total, or approximately 32,000 (N_a) excluding categories probably not relevant to tourists (e.g., community and government services, health care services, and several business categories such as financial). Approximate numbers for other locations are (all POIs in the area/probable tourist POIs or N_a): Amsterdam (the Netherlands) 110k/14k, Glasgow (UK) 47.7k/8.1k, Athens (Greece) 26.7k/4.3k, and Xanthi (Greece) 148/129. Let us assume a very active user who has rated 1k POIs (N_r). For the content-based methods, pre-processing takes an order of 1k operations, and 1 operation per update. Suggesting POIs for the city of Glasgow would take 8.1m operations with WkNN, while only 8.1k with Rated Rocchio. From the ratios of tourist POIs to all POIs in most of the above cities, it seems reasonable to assume a number of tourist global POIs of 15% of the 100 million, i.e., 15m (N). Assuming a—rather average in size, nowadays—CF service which serves 1m users (M), pre-processing and updating take 1 billion operations each, and suggesting takes 540k operations.

In summary, the method most suitable to be run at the user-side, e.g., in a user's mobile device, is Rated Rocchio, while for less active users (smaller N_r's, e.g. a few tens) WkNN may also be considered. Concerning CF, while it may be capable of providing suggestions in times comparable to or faster than WkNN, its pre-processing and updating times are rather prohibited for mobile devices, so it should be run at some provider.

Concerning the fusion methods, the ones we propose can be calculated fast in $O(N_a)$, thus they are limited by the most costly individual method combined. In this respect, a WkNN–Rated Rocchio combination would be possible, since it has effectiveness benefits.

We have not considered network use, yet it is easy to deduce that content-based methods are more demanding on network traffic, although still feasible, since they transfer information, e.g., for a few thousands of POIs, times a few tens of data pieces (e.g., keywords) per POI. Furthermore, we have not considered the size of the POI representation space, i.e., a few tens or hundreds of dimensions (e.g., keywords), which would multiply and linearly increase the running complexities. Note, however, that in the case of Rated Rocchio, instead of all POIs in the area of interest a provider could supply a pre-built index of them, further reducing the suggestion complexity and network traffic (indices are smaller than their corresponding collections) at the user's device.

5.3 Privacy

So far, we have investigated and identified the privacy leaks of the proposed methods, considering their straightforward implementation, i.e., without taking any special privacy-preserving steps. The rightmost column of Table 3 summarizes these privacy leaks. Expectedly, CF has the worst privacy characteristics since—as argued in Section 3.3—all users' ratings must be available at the recommender, as well as the location of the user at the time of request.

[19] www.factual.com/products/global.

Note that the choice of running a CF recommender at some remote centralized service rather than on users' devices is currently enforced by computational complexity rather than privacy. In principle, users can share ratings directly with each other, in a private way, by using, e.g., peer-to-peer connections over some anonymized network infrastructure such as Tor.[20] Consequently, private CF without the need of a provider would be possible on individual user devices, if the computational complexity was not prohibiting.

The privacy issue of CF when using remote centralized services, has been identified and dealt with by others before. In [35], the authors propose a new measure of similarity which is derived from the concordant, discordant, and tied pairs of ratings between two users. These are estimated by comparing each user's rating vector with a randomly generated one, extracting the aforementioned counts for both cases and estimating the corresponding counts between the two users. Like that, the service acquires pairwise similarity scores between the users without the need to break their privacy. Many others, e.g., [16, 39, 45], propose several different techniques of introducing statistical noise in the original user ratings while still being able to make useful calculations which allow the extraction of recommendations. In [15], the authors employ factor analysis and create a linear probabilistic model which is then encrypted. The update process is reduced to an iteration based on vector addition which is made private using homomorphic encryption functions. Each user's recommendations are extracted from the encrypted model through a global key which is shared in parts between the users.

All the above show that there are several solutions one could incorporate in a remote recommender in order to offer user privacy. Similarly, there exist solutions for content-based and other recommenders. For example, [2] propose a hybrid system that combines content-based, demographics, and CF techniques. They approach the privacy problem by splitting customer data between the merchant and a semi-trusted third party, using encryption and exchange of session keys between the parties. This process allows neither party to derive sensitive user information as the system isolates the user data from the user ID.

As shown in Table 3 and argued in Sections 3.1 and 3.2, our proposed content-based methods leak only the general location of the user in order to receive from some provider the most up-to-date information on the available POIs in the area of interest; this could be tolerated by privacy-sensitive users. In any case, if better privacy is required, the information on the general location of the user could be requested and received using again some anonymized network infrastructure, e.g., Tor, as mentioned above.

Concerning the fusion methods, the privacy leaks are determined by the "weakest link," thus any combination which includes a straightforward implementation of CF would have the worst privacy leaks and should be avoided for privacy-sensitive users.

Note that most, if not all, privacy-enhancing solutions mentioned above, assume co-operative providers who are willing to offer user privacy, often at their cost of a less effective user profiling for marketing purposes. Some related work dealing with querying uncooperative search engines in a private way is presented in [6, 7], which seems feasible to be adapted for the content-based recommendation methods.

6 CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH

Recommending venues or POIs is a hot topic in recent years, especially for tourism applications and mobile users. In this work, we have taken an effectiveness, efficiency, and privacy perspective of several suggestion methods. While the most important feature of such methods is effectiveness (users do not care for fast but bad results) which has been routinely evaluated by the IR and

[20] www.torproject.org.

RecSys communities with controlled experiments, we have also considered methods' computational complexities and privacy characteristics. Considering that higher levels of privacy can be achieved by loading users' mobile devices with privacy-sensitive parts of suggestion systems, the efficiency of suggestion methods becomes important. In this article, we have proposed, evaluated for their feasibility and effectiveness, and analyzed for their efficiency and privacy, several methods, as well as their combinations.

The first method is based on a W*k*NN classifier. We issue a candidate venue as a query to an index of all user's rated venues, and predict its rating by taking the weighted average of the ratings for the *k* semantically most similarly rated venues; here, we have tried several weighting methods. The second method is based on a Rocchio-like method we have developed for multi-graded relevance environments, which we call Rated Rocchio. Using a user's rated venues as training examples, the method builds a personalized query for the user. The Rated Rocchio query is then run on an index with the POIs in the area of interest in order to rank candidates. The aforementioned two content-based methods have been compared to standard methods for CF and variations, which consider the ratings of all users for all POIs, together. We have then investigated methods for fusing the results of all or some of the above. In this respect, rank-based fusion, such as Borda Count and Condorcet, has been employed, as well as rating-based fusion. The latter requires mapping retrieval scores or re-normalizing ratings of methods which do not produce ratings or compatible ratings, such as the Rated Rocchio and CF, respectively.

All the above have been evaluated for their effectiveness in two standard benchmark datasets, these provided and used by TREC's Contextual Suggestion Tracks in 2015 and 2016. Concerning the effectiveness of the proposed methods, we have reached the following conclusions.

- All individual content-based methods are suitable and effective for the task, with statistically insignificant differences between them. Compared to the state-of-the-art, e.g., median/best TREC runs, they achieve from above-median performance to the best performance, depending on the dataset and measures one looks at.
- CF suffers from data sparsity in the current datasets; only the user–user similarity variant has been possible to run in only one of the datasets (2015). Even then, although it is found to perform similarly to the content-based methods, we consider this inconclusive; its good performance seems to be accidental, benefiting from the very high quality candidate POIs in that dataset and narrow cut-offs (early-precision sensitivity) in the evaluation measures (see discussion in Section 5).
- Rank-based merging methods such as Borda Count or Condorcet are "safe" or robust options which yield at least similar or better effectiveness to the individual methods combined or, in roughly half of the cases, significantly better (e.g., three-way merges and two-way W*k*NN–RRocchio). The more methods combined, the better the effectiveness.
- Rating-based merging methods are even better than rank-based ones, when individual methods that produce ratings by nature are combined (e.g., CF–W*k*NN). Our attempts so far to map scores to ratings in the RRocchio method have produced inconsistent but promising results: significantly worse performance in one dataset and a solid better performance in the other.
- Merging seems to help more in promoting the rank of the first relevant result, "sharpening" the ranking, which could be really useful for users who spare time to visit a single POI.

Furthermore, a deeper analysis of the behavior of the methods has indicated possible limitations of the benchmark datasets used. Beyond their unsuitability for collaborative filtering experimentation, they also seem incapable of discriminating between good runs/systems, introducing some ceiling in the effectiveness that is possibly achievable. This also has to do with the early-precision

sensitivity of the evaluation measures; it is widely known that such measures are also notably unstable.

Concerning efficiency and privacy, we have theoretically argued and shown that the content-based methods can be efficiently implemented and run at the user's mobile device, with the minimum information leakage, this of the area of interest or user's location. This can be considered an acceptable leak, tolerated by users. Rated Rocchio is the most efficient of the two. While W*k*NN and CF could provide suggestions in comparable speeds in some cases, the latter has much more costly updates. Furthermore, CF is the worst of all methods privacy-wise, since all user preferences of all system users are leaked. Obviously, the efficiency and privacy of the fused methods are determined by the slowest method combined and the "weakest link" in privacy, respectively.

In summary, we believe that our findings can be very useful for developing effective operational systems, respecting user privacy. Especially, some dead-end, ineffective, or promising but less effective experiments we have also described, can be proven useful for revealing ground-breaking ways in future research. Further improvements could come from the following.

- —In the W*k*NN method, the tf-idf score which is suitable for ranking, may not be the best choice as a semantical distance measure—in this respect, other mechanisms could be explored.
- —In the Rated Rocchio method, we have mainly 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. In this respect, alternative centroid weightings could be investigated.
  In this work, we have already tried some alternatives for both of the above points, but with no significant improvements so far; these matters, however, seem to deserve a further investigation.
- —In fusing systems with rating-based methods, systems which do not produce ratings directly, such as Rated Rocchio, require some procedure for mapping scores to ratings. Since we have so far provided some baseline methods but with an inconsistent behavior across datasets, this matter may be considered open.
- —Independently from the above points, context could be enriched to include, for example, temporal information (time, day, season of the year, etc), weather conditions, or other contextual information [1, 40]. Beyond user-dependent features, some others consider venue-dependent features collected from location-based social networks [25] such as number of check-ins, number of likes, number of comments, number of photos, average Foursquare rating, and unique number of users. Surprisingly, they find that such features are more effective than user-dependent features for estimating the relevance of a suggestion, despite the fact that task is a personalized task.

Lastly, our experiments have indicated that more and/or better benchmark datasets would be welcome, especially ones that can facilitate experiments in both content-based and collaborative fashion. Also, the use of additional evaluation measures focusing on longer parts of rankings is recommended; although these may not be as suitable for capturing mobile user models, they are suitable for desktop users and maybe other applications in the future, and furthermore they may better discriminate systems.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-aware recommender systems. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer, 217–253. DOI: http://dx.doi.org/10.1007/978-0-387-85820-3_7

[2] Esma Aïmeur, Gilles Brassard, José Manuel Fernandez, Flavien Serge Mani Onana, and Zbigniew Rakowski. 2008. Experimental demonstration of a hybrid privacy-preserving recommender system. In *Proceedings of the The 3rd International Conference on Availability, Reliability and Security (ARES'08)*. IEEE Computer Society, 161–170. DOI: http://dx.doi.org/10.1109/ARES.2008.193

[3] Mohammad Aliannejadi, Seyed Ali Bahrainian, Anastasia Giachanou, and Fabio Crestani. 2015. University of lugano at TREC 2015: Contextual suggestion and temporal summarization tracks (see [55]). http://trec.nist.gov/pubs/trec24/papers/USI-CXTS.pdf.

[4] N. S. Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185. DOI: http://dx.doi.org/10.1080/00031305.1992.10475879 arXiv:http://www.tandfonline.com/doi/pdf/10.1080/00031305.1992.10475879

[5] Avi Arampatzis. 2001. Unbiased S-D threshold optimization, initial query degradation, decay, and incrementality, for adaptive document filtering. In *Proceedings of the 10th Text REtrieval Conference (TREC'01)*, Ellen M. Voorhees and Donna K. Harman (Eds.), Vol. Special Publication 500-250. National Institute of Standards and Technology (NIST). http://trec.nist.gov/pubs/trec10/papers/KUN-TREC10.pdf.

[6] Avi Arampatzis, George Drosatos, and Pavlos S. Efraimidis. 2015. Versatile query scrambling for private web search. *Information Retrieval Journal* 18, 4 (2015), 331–358. DOI: http://dx.doi.org/10.1007/s10791-015-9256-0

[7] Avi Arampatzis, Pavlos S. Efraimidis, and George Drosatos. 2013. A query scrambler for search privacy on the internet. *Information Retrieval* 16, 6 (2013), 657–679.

[8] Javed A. Aslam and Mark H. Montague. 2001. Models for metasearch. In *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel (Eds.). ACM, 275–284. DOI: http://dx.doi.org/10.1145/383952.384007

[9] Sandeep Avula, John O'Connor, and Jaime Arguello. 2013. A nearest neighbor approach to contextual suggestion (s [54]). http://trec.nist.gov/pubs/trec22/papers/unc-context.pdf.

[10] L. Barkhuus and A. Dey. 2003. Location-based services for mobile telephony: A study of user's privacy concerns. In *Proceedings of the 9th IFIP TC13 International Conference on Human-Computer Interaction (Interact'03)*. IOS Press, Zürich, Switzerland.

[11] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46 (2013), 109–132.

[12] Jean-Charles de Borda. 1784. Mémoire sur les élections au scrutin, Histoire de lAcadémie royale des sciences pour 1781. *Paris (English translation by De Grazia, A., 1953. Isis 44)* (1784).

[13] Joan Borràs, Antonio Moreno, and Aida Valls. 2014. Intelligent tourism recommender systems: A survey. *Expert Systems with Applications* 41, 16 (2014), 7370–7389.

[14] Matthias Braunhofer, Mehdi Elahi, Francesco Ricci, and Thomas Schievenin. 2013. *Context-Aware Points of Interest Suggestion with Dynamic Weather Data Management*. Springer International Publishing, Cham, 87–100. DOI: http://dx.doi.org/10.1007/978-3-319-03973-2_7

[15] John F. Canny. 2002. Collaborative filtering with privacy via factor analysis. In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Kalervo Jävelin, Micheline Beaulieu, Ricardo A. Baeza-Yates, and Sung-Hyon Myaeng (Eds.). ACM, 238–245. DOI: http://dx.doi.org/10.1145/564376.564419

[16] Fran Casino, Josep Domingo-Ferrer, Constantinos Patsakis, Domenec Puig, and Agusti Solanas. 2015. A k-anonymous approach to privacy preserving collaborative filtering. *Journal of Computer and System Sciences* 81, 6 (2015), 1000–1011. DOI: http://dx.doi.org/10.1016/j.jcss.2014.12.013

[17] Manajit Chakraborty, Hitesh Agrawal, Himanshu Shekhar, and C. Ravindranath Chowdary. 2015. Contextual suggestion using tag-description similarity (see [55]). http://trec.nist.gov/pubs/trec24/papers/DPLAB_IITBHU-CX.pdf.

[18] Weitong Chen, Hanchen Li, and Zhen Yang. 2015. BJUT at TREC 2015 contextual suggestion track (see [55]). http://trec.nist.gov/pubs/trec24/papers/BJUT-CX.pdf.

[19] Nicolas de Condorcet. 1785. Essai sur l'application de l'analyse la probabilitié des décisions rendues à la pluralité des voix. *Paris: l'Imprimerie Royale; Translated in Mclean and Urken 1995* (1785), 91–113.

[20] George Danezis, Josep Domingo-Ferrer, Marit Hansen, Jaap-Henk Hoepman, Daniel Le Métayer, Rodica Tirtea, and Stefan Schiffner. 2015. Privacy and data protection by design—From policy to engineering. *CoRR* abs/1501.03726 (2015). http://arxiv.org/abs/1501.03726

[21] Alexandre De Spindler, Moira C. Norrie, Michael Grossniklaus, and Beat Signer. 2006. Spatio-temporal proximity as a basis for collaborative filtering in mobile environments. In *Proceedings of the CAISE Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS'06)*.

[22] Adriel Dean-Hall, Charles L. A. Clarke, Jaap Kamps, Julia Kiseleva, and Ellen M. Voorhees. 2015. Overview of the TREC 2015 contextual suggestion track (see [55]). Retrieved from http://trec.nist.gov/pubs/trec24/papers/Overview-CX.pdf.

[23] Adriel Dean-Hall, Charles L. A. Clarke, Jaap Kamps, and Paul Thomas. 2013. Evaluating contextual suggestion. In *Proceedings of the 5th International Workshop on Evaluating Information Access (EVIA'13)*, Ruihua Song and William Webber (Eds.). National Institute of Informatics (NII). Retrieved from http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings10/pdf/EVIA/09-EVIA2013-DeanHallA.pdf.

[24] Adriel Dean-Hall, Charles L. A. Clarke, Nicole Simone, Jaap Kamps, Paul Thomas, and Ellen M. Voorhees. 2013. Overview of the TREC 2013 contextual suggestion track (see [54]). Retrieved from http://trec.nist.gov/pubs/trec22/papers/CONTEXT.OVERVIEW.pdf.

[25] Romain Deveaud, M-Dyaa Albakour, Craig Macdonald, and Iadh Ounis. 2014. On the importance of venue-dependent features for learning to rank contextual suggestions. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM'14)*, Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang (Eds.). ACM, 1827–1830. DOI: http://dx.doi.org/10.1145/2661829.2661956

[26] George Drosatos, Pavlos S. Efraimidis, Avi Arampatzis, Giorgos Stamatelatos, and Ioannis N. Athanasiadis. 2015. Pythia: A privacy-enhanced personalized contextual suggestion system for tourism. In *Proceedings of the 39th Annual IEEE Computers, Software and Applications Conference (COMPSAC'15)*. IEEE Computer Society, 822–827. DOI: http://dx.doi.org/10.1109/COMPSAC.2015.88

[27] George Drosatos, Giorgos Stamatelatos, Avi Arampatzis, and Pavlos S. Efraimidis. 2013. DUTH at TREC 2013 contextual suggestion track (see [54]). Retrieved from http://trec.nist.gov/pubs/trec22/papers/DUTH-context.pdf.

[28] Pavlos S. Efraimidis, George Drosatos, Avi Arampatzis, Giorgos Stamatelatos, and Ioannis N. Athanasiadis. 2016. A privacy-by-design contextual suggestion system for tourism. *Journal of Sensor and Actuator Networks* 5, 2 (2016), 10. DOI: http://dx.doi.org/10.3390/jsan5020010

[29] European Parliament. 1995. Directive 95/46/EC. In *Official Journal L 281*. 0031–0050. Retrieved April 20, 2016 from http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:HTML.

[30] Damianos Gavalas and Michael Kenteris. 2011. A web-based pervasive recommendation system for mobile tourist guides. *Personal and Ubiquitous Computing* 15, 7 (2011), 759–770.

[31] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. 2014. Mobile recommender systems in tourism. *Journal of Network and Computer Applications* 39, 0 (2014), 319–333.

[32] Seyyed Hadi Hashemi, Charles L. A. Clarke, Jaap Kamps, Julia Kiseleva, and Ellen M. Voorhees. 2016. Overview of the TREC 2016 contextual suggestion track (see [56]). Retrieved from http://trec.nist.gov/pubs/trec25/papers/Overview-CS.pdf.

[33] Seyyed Hadi Hashemi, Mostafa Dehghani, and Jaap Kamps. 2015. Parsimonious user and group profiling in venue recommendation (see [55]). Retrieved from http://trec.nist.gov/pubs/trec24/papers/UAmsterdam-CX.pdf.

[34] Georgios Kalamatianos and Avi Arampatzis. 2016. Recommending points-of-interest via weighted *k*NN, rated rocchio, and borda count fusion (see [56]). Retrieved from http://trec.nist.gov/pubs/trec25/papers/DUTH-CX.pdf.

[35] Neal Lathia, Stephen Hailes, and Licia Capra. 2007. Private distributed collaborative filtering using estimated concordance measures. In *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys'07)*, Joseph A. Konstan, John Riedl, and Barry Smyth (Eds.). ACM, 1–8. DOI: http://dx.doi.org/10.1145/1297231.1297233

[36] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[37] Jarana Manotumruksa, Craig MacDonald, and Iadh Ounis. 2016. Modelling user preferences using word embeddings for context-aware venue recommendation. *CoRR* abs/1606.07828 (2016).

[38] Richard McCreadie, Saul Vargas, Craig MacDonald, Iadh Ounis, Stuart Mackie, Jarana Manotumruksa, and Graham McDonald. 2015. University of Glasgow at TREC 2015: Experiments with terrier in contextual suggestion, temporal summarisation and dynamic domain tracks (see [55]). Retrieved from http://trec.nist.gov/pubs/trec24/papers/uogTr-CXTSDD.pdf.

[39] Frank McSherry and Ilya Mironov. 2009. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki (Eds.). ACM, 627–636. DOI: http://dx.doi.org/10.1145/1557019.1557090

[40] Kevin Meehan, Tom Lunney, Kevin Curran, and Aiden McCaughey. 2013. Context-aware intelligent recommendation system for tourism. In *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 328–331.

[41] Jian Mo, Luc Lamontagne, and Richard Khoury. 2015. Laval University and Lakehead University experiments at TREC 2015 contextual suggestion track (see [55]). Retrieved from http://trec.nist.gov/pubs/trec24/papers/LavallVA-CX.pdf.

[42] Rabia Nuray and Fazli Can. 2006. Automatic ranking of information retrieval systems using data fusion. *Information Processing & Management* 42, 3 (2006), 595–614. DOI: http://dx.doi.org/10.1016/j.ipm.2005.03.023

[43] Makbule Gulcin Ozsoy. 2016. From word embeddings to item recommendation. *CoRR* abs/1601.01356 (2016). http://arxiv.org/abs/1601.01356

[44] Konstantinos Pliakos and Constantine Kotropoulos. 2014. Simultaneous image clustering, classification and annotation for tourism recommendation. In *Proceedings of the 8th Hellenic Conference on Artificial Intelligence: Methods and Applications (SETN 2014)*, Lecture Notes in Computer Science, Vol. 8445, Aristidis Likas, Konstantinos Blekas, and Dimitris Kalles (Eds.). Springer, 630–640.

[45] Huseyin Polat and Wenliang Du. 2003. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03)*. IEEE Computer Society, 625–628. DOI: http://dx.doi.org/10.1109/ICDM.2003.1250993

[46] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2012. *Mining of Massive Datasets*. Vol. 1. Cambridge University Press, Cambridge.

[47] Ioannis Refanidis, Christos Emmanouilidis, Ilias Sakellariou, Anastasios Alexiadis, Remous-Aris Koutsiamanis, Konstantinos Agnantis, Aimilia Tasidou, Fotios Kokkoras, and Pavlos S. Efraimidis. 2014. myVisitPlanner^{GR}: Personalized itinerary planning system for tourism. In *Proceedings of the 8th Hellenic Conference on Artificial Intelligence: Methods and Applications (SETN'14)*, Lecture Notes in Computer Science, Vol. 8445, Aristidis Likas, Konstantinos Blekas, and Dimitris Kalles (Eds.). Springer, 615–629.

[48] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender Systems Handbook*. Springer, 1–35.

[49] J. J. Rocchio. 1971. Relevance feedback in information retrieval. In *The Smart Retrieval System—Experiments in Automatic Document Processing*, G. Salton (Ed.). Prentice-Hall, Englewood Cliffs, NJ, 313–323.

[50] Mark D. Smucker and Charles L. A. Clarke. 2012. Time-based calibration of effectiveness measures. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'12),*, William R. Hersh, Jamie Callan, Yoelle Maarek, and Mark Sanderson (Eds.). ACM, 95–104. DOI: http://dx.doi.org/10.1145/2348283.2348300

[51] S. Stabb, H. Werther, F. Ricci, A. Zipf, U. Gretzel, D. R. Fesenmaier, C. Paris, and C. Knoblock. 2002. Intelligent systems for tourism. *IEEE Intelligent Systems* 17, 6 (Nov. 2002), 53–66.

[52] Aidan Trees, Kevin Danaher, Zach Siatkowski, Darren Lim, and Tom Heritage. 2015. Siena College's institute of Artificial Intelligence TREC 2015 contextual suggestion track (see [55]). Retrieved from http://trec.nist.gov/pubs/trec24/papers/Siena_SUCCESS-CX.pdf.

[53] Sofia Tsekeridou, Vassileios Tsetsos, Aimilios Chalamandaris, Christodoulos Chamzas, Thomas Filippou, and Christos Pantzoglou. 2014. iGuide: Socially-enriched mobile tourist guide for unexplored sites. In *Proceedings of the 8th Hellenic Conference on Artificial Intelligence: Methods and Applications (SETN'14),* Lecture Notes in Computer Science, Vol. 8445, Aristidis Likas, Konstantinos Blekas, and Dimitris Kalles (Eds.). Springer, 603–614.

[54] Ellen M. Voorhees (Ed.). 2013. *Proceedings of the 22nd Text REtrieval Conference (TREC'13)*. Vol. Special Publication 500-302. National Institute of Standards and Technology (NIST). Retrieved from http://trec.nist.gov/pubs/trec22/trec2013.html.

[55] Ellen M. Voorhees and Angela Ellis (Eds.). 2015. *Proceedings of the 24th Text REtrieval Conference (TREC'15)*. Vol. Special Publication 500-319. National Institute of Standards and Technology (NIST). Retrieved from http://trec.nist.gov/pubs/trec24/trec2015.html.

[56] Ellen M. Voorhees and Angela Ellis (Eds.). 2016. *Proceedings of the 25h Text REtrieval Conference (TREC'16)*. Vol. Special Publication 500-321. National Institute of Standards and Technology (NIST). Retrieved from http://trec.nist.gov/pubs/trec25/trec2016.html.

[57] Yuan Wang, Jie Liu, Yalou Huang, Yongfeng Zhang, Yi Zhang, and Xintong Zhang. 2015. Exploration of semantic-aware approach for contextual suggestion using knowledge from the open web (see [55]). http://trec.nist.gov/pubs/trec24/papers/ucsc-CX.pdf.

[58] Christopher Wing and Hui Yang. 2014. FitYou: Integrating health profiles to real-time contextual suggestion. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'14)*, Shlomo Geva, Andrew Trotman, Peter Bruza, Charles L. A. Clarke, and Kalervo Järvelin (Eds.). ACM, 1263–1264. DOI: http://dx.doi.org/10.1145/2600428.2611185

[59] Zheng Xiang and Iis Tussyadiah. 2014. *Information and Communication Technologies in Tourism 2014*. Springer, New York.

[60] Peilin Yang and Hui Fang. 2015. University of Delaware at TREC 2015: Combining opinion profile modeling with complex context filtering for contextual suggestion (see [55]). http://trec.nist.gov/pubs/trec24/papers/udel_fang-CX.pdf.

[61] Wan-Shiou Yang and San-Yih Hwang. 2013. iTravel: A recommender system in mobile peer-to-peer environment. *Journal of Systems and Software* 86, 1 (2013), 12–20.

[62] Lina Yao, Quan Z. Sheng, Yongrui Qin, Xianzhi Wang, Ali Shemshadi, and Qi He. 2015. Context-aware point-of-interest recommendation using tensor factorization with social regularization. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Ricardo A. Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto (Eds.). ACM, 1007–1010. DOI: http://dx.doi.org/10.1145/2766462.2767794

[63] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. 2013. Time-aware point-of-interest recommendation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'13)*, Gareth J. F. Jones, Paraic Sheridan, Diane Kelly, Maarten de Rijke, and Tetsuya Sakai (Eds.). ACM, 363–372. DOI: http://dx.doi.org/10.1145/2484028.2484030

[64] J. Zhan, Chia-Lung Hsieh, I-Cheng Wang, Tsan-Sheng Hsu, Churn-Jung Liau, and Da-Wei Wang. 2010. Privacy-preserving collaborative recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 40, 4 (July 2010), 472–476.

Received January 2017; revised May 2017; accepted July 2017