Online evaluation of point-of-interest recommendation systems

Adriel Dean-Hall
University of Waterloo

Charles L. A. Clarke
University of Waterloo

Jaap Kamps
University of Amsterdam

Julia Kiseleva
Eindhoven University of Technology

ABSTRACT

In this work we describe a system to evaluate multiple point-of-interest recommendation systems. In this system each recommendation service will be exposed online and crowdsourced assessors will interact with merged results from multiple services, which are responding to suggestion requests live, in order to determine which system performs best. This work builds upon work done previously as part of the TREC Contextual Suggestion Track and describes plans for how the track will be run in 2015.

1. INTRODUCTION

Many point-of-interest recommendation systems have been developed, each using different techniques for making recommendations. Often, when these systems are evaluated, they are operating on different datasets or different factors are compared. Having a framework to fit such systems into and be able to compare them using fair, standardized techniques will help us determine which systems performed the best.

The TREC Contextual Suggestion track [6] has been running for three years since 2012. In this track systems, which provide personalized point-of-interest suggestions, are designed. The past three iterations of the track have followed closely with the traditional TREC evaluation methodology: participants are given topics and develop a set of results for each topic, the results are then evaluated by assessors and scores are assigned to each participating system based on these judgements [11]. Specifically, a set of profiles (ratings for a set of attractions) and contexts (names of cities) were released to participants. For each profile+context pair participants returned a set of ranked suggestions. These suggestions were then judged by the assessors who originally created the profiles and a score was assigned to each participant's set of results.

One disadvantage of this setup is that, for this track, topics are actually *personal* preferences provided by crowdsourced assessors who, after providing their preferences, have to wait weeks for attraction suggestions. This wait makes the task of assessing more difficult as judgement is broken over long period and assessors have to remember previous interactions with the system. Also, the longer the wait, the more difficult it is to get crowdsourced assessors to return to the task.

In this article we will describe a setup that allows for attraction recommendation services to be compared with users issuing requests where suggestions are made live. Here participating recommendation services will have an online system which is able to respond to a suggestion request immediately. When a user (or an assessor) is ready for suggestions they make a request. The system will then send that user's profile and the name of the city to services. Suggestions will be made by multiple recommendation services and the returned results will be merged and presented to the user. The user will then interact with the results which provides feedback on how good each service's suggestions are. A score for each service is continuously updated until the experiment ends.

We will describe the interface recommendation services need to implement in order to participate, how they will be compared, and some challenges that moving from a batch-style to live evaluation setup introduces.

2. RELATED WORK

Point-of-interest recommendation is an area several researchers are perusing. Braunhofer et al. [5] worked on an application that made personalized recommendations within cities; Adomavicius et al. [1] used collaborative filtering for similar goals incorporating temporal features; Baltrunas et al. [3] used information such as budget and familiarity with the area. Additionally several systems have been developed as part of the Contextual Suggestion track including a system that used textual similarity between attractions [8], and a systems that found reviews to be an informative feature [12]. The goal is to bring the efforts of all these systems into one framework in order to compare them fairly.

In addition to this framework being built upon the Contextual Suggestion track our work is also inspired by work done during the plista dataset challenge [7]. In these experiments multiple competing systems registered to make recommendation about related articles users might find interesting based on the article they were currently reading and previous interactions with the system. These recommendations had to be made live as they were presented to users while they were browsing news articles. Another

ECIR Supporting Complex Search Task Workshop '15 Vienna, Austria
Published on CEUR-WS: `http://ceur-ws.org/Vol-1338/`.

source of inspiration are challenges such as the Netflix prize [4] and various Kaggle competitions[1], which, while not typically evaluated live, make use of various techniques, e.g., leaderboards, in order to provide feedback to participants.

3. SERVICE INTERFACE

In order to develop a framework for testing point-of-interest recommendation services we need to determine an interface that users will use to communicate with them. As parameters for each recommendation request the systems take in the user's profile (see Section 4 for a description of the profile) and, as context, the city which the user wants recommendations for. We could also gather more contextual information about our users during each request, for example, a more precise location, who the user is travelling with (family, friends, alone), etc. However, currently, we only use the city and profile as input. The result returned to users will consist of an ordered list of attractions that the service thinks the user will like.

This is similar to the information participants in previous iterations of the track had available to them but instead of getting a batch of profiles and cities and returning a batch of results, services will recieve a single profile and city for each request. Additionally, instead of having a fixed profile for each user, on each request the profile may be updated with liked suggestions from previous requests.

4. USER PROFILES

This leads us to the question of what a user's profile consists of. Initially, for a new user, no information is in their profile. As the user asks for suggestions and interacts with them their profile will expand. For each attraction that has been recommended to the user the data about their interaction is added to their profile. Examples of interaction include the user viewing the attraction's website, the user "starring" the attraction, and the user rating the attraction. Currently, these three pieces of information are recorded for each attraction the user has interacted with, however other interactions, e.g., reviews written, could also be included in the profile. As the user interacts with the system their profile will expand giving recommendation services a better opportunity to make more personalized suggestions.

Once interaction data from an attraction has been added to the user's profile it is essentially available publicly to all services. One issue with this setup is that certain requests will have small profiles and certain requests will have larger profiles. Limiting the size of the profile will allow us not to worry as much about how the size of the profile affects service performance. One option to resolve this is, instead of adding every piece of interaction to a user's profile, only add certain attractions. One simple method of doing this is to only add attraction interaction data for a subset of cities. This will also allow us to ask for suggestions for the same city multiple times (if that city is not part of the profile) and necessarily not have to return to users for result interactions.

Another potential feature is to push any updates to the profiles to services. This will allow them to get feedback into how well they are performing and whether the suggestions they are making are actually liked by users without having to pool profiles or wait for another request from the same user. Services can then update their strategies and attempt to improve suggestion results for future requests.

[1] http://www.kaggle.com/

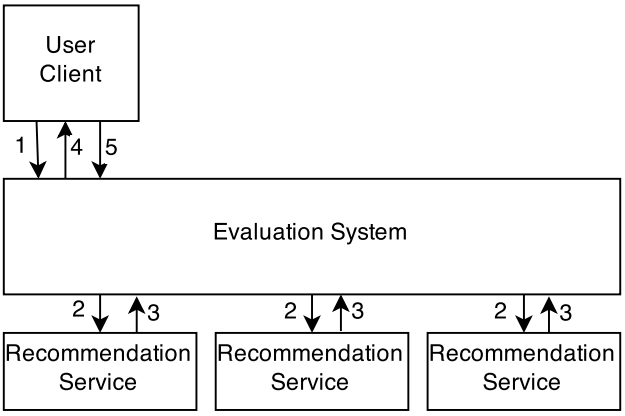

Figure 1: 1. Request sent; 2. System passes request to services; 3. Services respond with suggestions; 4. Suggestions are merged and sent to user; 5. User sends suggestion interactions to system.

5. DATASET COLLECTION

In previous iterations of this track services were allowed to recommend any attraction they found on the open web. For simplicity, the points-of-interest that services are allowed to recommend in this experiment come from a fixed collection of attractions. Services will simply return a list of attraction IDs. When they are displayed to users each attraction will consist of a title, short description, and website URL with more information about the attraction. Users will use this information to make a decision about whether they like a particular attraction. Again, here we are presenting this basic information about each attraction but additional information, such as the attraction's category or reviews about the attraction, could also be presented to users.

This pool of attractions is collected as part of an ongoing effort by several research groups who have expertise in dealing with gathering this sort of information due to participation in previous Contextual Suggestion TREC tracks. Having a fixed data collection will allow us to limit which attractions are suggested and will allow for greater reusability of the judgements provided by users.

6. SERVICE EVALUATION

So, in order to develop a recommendation service that fits into this framework services must set up a server that responds to suggestions requests with a list of attraction IDs. The goal of forcing services into this framework is so that we can compare the performance between multiple services. Instead of having users communicate directly with a recommendation service they will communicate with an intermediary system. Services will be required to register themselves with this system and the system will then pass recommendation requests to each service, logging the services' responses.

Each service will be given an opportunity to make suggestions. One option to do this is, for each request, select one of the services at random and present the results from

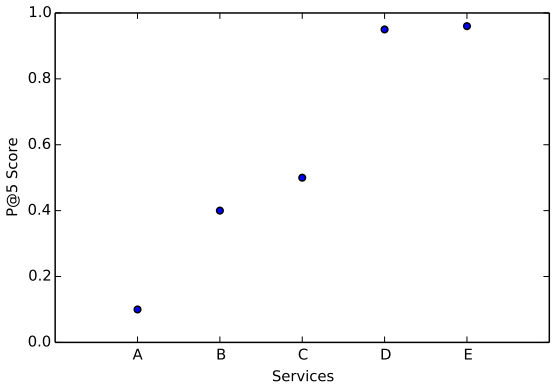

Figure 2: Potential scores given to five sytems at some point during the experiment.

the service to the user. The user will then interact with the system and based on this interaction we can determine how good the suggestions were. As more suggestion requests are made each service will be given multiple chances to make recommendations and services can be compared.

We take a slightly different approach where, for each suggestion request a subset of the services are queried and the results from all these services are merged into a final list of suggestions which is presented to users. If the user interacts with a suggestion the score of the service that made that suggestion will be affected. It is possible for multiple services to make the same suggestion, in this case if the user interacts with the suggestion all the services that made it will have their score affected.

This setup will give services more opportunities to make suggestions (during each request rather than only some), however we will need a method of merging requests from multiple services. Multiple result interleaving approaches have been discussed by Radlinski and Craswell which would be appropriate for our purposes [9], including the team draft method proposed by Radlinski et al. [10].

The reason a subset of the services rather than all services are selected is that, realistically, most users will only view the attractions near the top of the list. If we try to compare too many services the user may not see any suggestions from some of the services (or only see few suggestions from each service). In order to prevent this situation five services are chosen for each suggestion request (this number is chosen somewhat arbitrarily). We also limit the number of suggestions each service can make per request to 50.

7. SCORING

Again, we have our suggestion services which produce ranked lists of suggestions. Each suggestion request will be sent out to multiple services and user's will interact with a list of merged responses. The user's interaction with the suggestions will allow a score to be calculated for each service. For each point-of-interest interacted with we can calculate a usefulness score based on how highly the user rated it and whether the user visited the website or "starred" it. We also collect timing data for each session so we can incorporate how long users spent on each attraction into our scoring metric. Precision at rank k, mean reciprocal rank, and a modified version of time-biased gain have been used in previous iterations of this track and can be used here as well.

Since we are not involving each service in each suggestion request we need to choose which services to involve. The simplest way to choose is to pick services randomly, however we should keep our end goal in mind here. Our goal is to find the correct ordering of services in terms of performance. So, for example, after a certain amount of requests have been made, if a particular service performs much more poorly than any other service we are not likely to learn more information about the correct ordering of services if we pick it as often as other services. On the other hand, if two services have very similar performance it may be worthwhile to pick them more often in order to determine which of the two services perform better. In Figure 2 we already know that service A performs poorly and there is probably more to gain by comparing services B and C.

We should also note that we are only interested in telling the difference between two systems if they have enough of a difference between them. If one service performs better than another but an end user would not realistically be able to tell the difference between them then it is not worthwhile spending a bunch of resources determining their correct ordering. In Figure 2 services D and E perform so similarly that it is probably not worth comparing them.

We leave this issue of selecting services based on their current ranking for future work and for now simply select systems for each request randomly. It is worth nothing that we only expect a handful of services to register for this system initially and they can all probably be involved in every or most suggestion requests.

In previous iterations of this track services waited until the experiments were done to recieve feedback on how well they performed. An option being explored for this experiment is to provide scores or a leaderboard for services every so often so that services can see how well they are performing and use that feedback to improve their service throughout the experiment.

8. ASSESSORS

So far we have been discussing a system which allows users to interact with different recommendation services. Because we don't have an existing userbase to run these experiments on we will use paid assessors to interact with the system in a similar way to real users. In past iterations of this track we have found crowdsourced workers to be useful in these sorts of tasks. Additionally Ageev et al. were successful in simulating search interaction data with crowdsourced workers [2]. For this experiment we will solicit hundreds of crowdsourced workers to make suggestion requests and interact with the results. They will be asked to interact with the results based on their own personal preferences. Payment will be issued based on how many and for how long result lists are interacted with.

Additionally, once the system has been set up and services are registered and running, the setup can provide value to real users who can continue to use the system outside of the track experiments. This will allow services to continue to receive feedback on their performance even outside of TREC.

9. SERVICE EFFICIENCY

When a request is made the user is expecting a response within a short amount of time. Services will have to be always available and be able to respond quickly. Once requests have been sent out to services, if a response takes too long to be returned that service will not be given an opportunity to contribute to the final list of suggestions presented to the user. In order to help services maintain responsiveness they will be allowed to register multiple servers. For each request one of the service's servers will be chosen to respond to the request. This will provide some robustness to the system should a particular server become unavailable. Additionally we can optionally incorporate each service's respond time into their score and have efficiency influence the ordering of services.

As an additional fallback mechanism, in case no service responds to a particular request, a baseline service will be developed that is always available and responds to every request. If no service responds the results from the baseline service will be presented to users. This ensures that users always receive *some* response. This fallback mechanism will gather its results from a commercial web service.

10. PERSONALIZED DESCRIPTIONS

The goal of each service is to select points-of-interest that the service predicts the user will like. These suggestions' titles, descriptions, and URLs are displayed to the user. The descriptions about each attraction shown to users are generic descriptions for that attraction. Services may want to modify the descriptions slightly in order to include, for example, why this particular user may find the attraction interesting. Services will be given an opportunity to provide personalized descriptions for each attraction in order to include this kind of information. Evaluation for these descriptions will be done separately from the main evaluation into service performance.

In previous iterations of this track every service had to provide descriptions for all suggestions. The decision to make this an optional task was based on feedback that most services were simply providing generic descriptions, which in this experiment we are providing instead. Generating the generic descriptions ourselves will provide us with another point of standardization between services to allow for more fair comparisons.

11. USER INTERFACE

Currently the interface that users use to make suggestions requests and interact with service results is a web based interface. Users will select a city from a list and then be presented with merged results from multiple systems. This allows for crowdsourced assessors to easily provide system feedback. However, the system is designed so that other methods of presenting results to users could easily be used. In particular the API allows any developer to build a mobile application which enables users to interact with the system. Point-of-interest recommendation lends itself to mobile users and having multiple vectors for users to interact with the system is one of the future goals for this project.

The source for this project is currently available online: `https://github.com/akdh/entertain-me`.

12. CONCLUSION

We have briefly given an overview of a system that is used to evaluate multiple point-of-interest recommendation services live using crowdsourced workers. This system will be used to run the TREC 2015 Contextual Suggestion Track. This experiment differs from previous years because services will have to available online during the experiment, suggestions will have to be delivered live, and assessment and evaluation can be a lot more fluid. If you are interested in registering your service for this experiment you can find out more about participating on the TREC website[2] and the Contextual Suggestion Track website[3].

13. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[2] M. Ageev, Q. Guo, D. Lagun, and E. Agichtein. Find it if you can: A game for modeling different types of web search success using interaction data. In *Proceedings of ACM SIGIR*, 2011.

[3] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context relevance assessment and exploitation in mobile recommender systems. *Personal Ubiquitous Comput.*, 16(5):507–526, June 2012.

[4] J. Bennett and S. Lanning. The netflix prize. 2007.

[5] M. Braunhofer, M. Elahi, and F. Ricci. Usability assessment of a context-aware and personality-based mobile recommender system. In *E-Commerce and Web Technologies*, volume 188, pages 77–88. Springer, 2014.

[6] A. Dean-Hall, C. L. A. Clarke, J. Kamps, P. Thomas, and E. Voorhees. Overview of the TREC 2014 contextual suggestion track. In *Proceedings of TREC*, Gaithersburg, Maryland, 2014.

[7] B. Kille, F. Hopfgartner, T. Brodt, and T. Heintz. The plista dataset. In *Proceedings of ACM NRS*, 2013.

[8] D. Milne, P. Thomas, and C. Paris. Finding, weighting and describing venues: Csiro at the 2012 trec contextual suggestion track. In *Proceedings of TREC*, 2012.

[9] F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *Proceedings of ACM WSDM*, 2013.

[10] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 43–52, New York, NY, USA, 2008. ACM.

[11] E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. Digital Libraries and Electronic Publishing. The MIT Press, 2005.

[12] P. Yang and H. Fang. An opinion-aware approach to contextual suggestion. In *Proceedings of TREC*, 2013.

[2] `http://trec.nist.gov`

[3] `https://sites.google.com/site/treccontext/`