An Evaluation of Contextual Suggestion

by

Adriel Dean-Hall

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2014

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis examines techniques that can be used to evaluate systems that solve the complex task of suggesting points of interest to users. A traveller visiting an unfamiliar, foreign city might be looking for a place to have fun in the last few hours before returning home. Our traveller might browse various search engines and travel websites to find something that he is interested in doing, however this process is time consuming and the visitor may want to find some suggestion quickly.

We will consider the type of system that is able to handle this complex request in such a way that the user is satisfied. Because the type of suggestion one person wants will differ from the type of suggestion another person wants we will consider systems that incorporate some level of personalization. In this work we will develop user profiles that are based on real users and set up experiments that many research groups can participate in, competing to develop the best techniques for implementing this kind of system. These systems will make suggestion of attractions to visit in various different US cities to many users.

This thesis is divided into two stages. During the first stage we will look at what information will go into our user profiles and what information we need to know about the users in order to decide whether they would visit an attraction. The second stage will be deciding how to evaluate the suggestions that various systems make in order to determine which system is able to make the best suggestions.

Acknowledgements

I would like to thank my supervisor, Charlie Clarke, who introduced me to this field and provided me just the right amount of guidance over the past two years while giving me the freedom to chase after a couple butterflies. Charlie is a good friend and I am much better off knowing him.

I would also like to thank current and former members of the PLG lab: Azin, Bahareh, Younos, Gobi, Nomair, Gaurav, Adam, Luchen, Ashif, Brad, Maheedhar, and Jack. You guys made my time at Waterloo what it is.

A big thank you to my family, I am lucky to have such great parents and such an awesome brother. Thank you guys for always allowing me to be myself, not many people would put up with that guy.

Finally I would like to thank the hundreds of participants in my studies and the dozens of TREC participants who made this work possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Rome, Tokyo, Gaithersburg. Imagine a traveller visiting these cities. Perhaps our fictional traveller is taking a vacation and wants to get the most out of his trip as possible, or perhaps he has just finished attending a meeting and has a few hours to kill before leaving. Regardless of the situation, our traveller wants to find something entertaining to do.

Our traveller has a smartphone, a connection to the internet, and a laptop which he uses whenever he needs to browse the internet for longer than would be comfortable on his phone. He has used several online travel guides and business review sites in the past when finding something to do. He looks up the city he is in on Lonely Planet, Yelp, Google Places, Wikitravel, TripAdvisor, or another similar site. The basic service provided by all of these sites is a listing of things to do. Typically these sites have a description, some photos, reviews, contact details, and other information available for each attraction. Often listings are ordered by rating or popularity that way things that are more attractive appear at the top of the list. Attractions are sometimes grouped into categories, such as "parks" or "restaurants" and the sites typically provide tools that can be used to filter attractions based on category. Some services offer a way of entering keywords that allow the set of listings to be narrowed down to just attractions that are related to those keywords.

Our traveller has his favourite travel site for finding attractions which he navigates to, spends a few minutes or longer searching for a handful of things he thinks he might be interested in, puts the information in his phone, and then heads out to explore the city. The question that we explore in this thesis is: How can we improve upon this process? We want a system that shows users how to have a good time. The ideal is that our traveller pulls out his smartphone, clicks a button, and a list of suggestions that the traveller finds interesting appears. Note that this list is not the same list that would be generated from

one of the above services. In the second case the attraction that the traveller finds most interesting appears at the very top of the list, the second item in the list is the second most interesting attraction, etc. In the first case the traveller enters a city they are in (this is sometimes automatically detected) and is presented with a list which he then has to filter and search through (both manually and by using the tools of the service when available) in order to find the attractions that are most interesting to him.

One of the key aspects of this thesis is personalization. We consider whether providing a list of attractions tailored specifically to our traveller would get us closer to our magic button which delivers a perfect list of attractions. Another important aspect that we will consider is localization. In this thesis we will investigate is how incorporating user location and time can bring us closer to our goal.

1.1 Research Approach

If we want to tailor attraction listings to individuals we need to know something about the user that allows us predict which attractions will be the most interesting to him. For our experiments we make the observation that our traveller will have some favourite places that he likes to go to in his own city. He will have a favourite restaurant, a favourite place to hangout on Friday nights, etc. If we know what attractions our traveller likes (and dislikes) in his home city then we can use this information to predict what he will like in other cities. The information that we know about our traveller makes up a profile.

Real systems would be able to gather data about users from, for example, what kinds of searches a user does and what kinds of websites a user visits (and how long the user visits the website for). It could also be gathered, more directly, from a check-in type system like Foursquare or reviews that the user has written on a site like Yelp. When deciding on what information will be contained in a profile we will keep in mind real systems and come up with profiles that such systems could reasonably gather.

Another aspect that we want to consider when developing profiles is privacy. An extremely privacy sensitive system would gather no data about its users whereas a system not concerned with privacy whatsoever might gather data about everywhere the user has been, who they meet, what they did, etc. When developing a profile we will gather as little data as we need to on our users while still providing a useful service. If we can create a system that works with a minimal profile then we can create a system that is privacy sensitive.

For our experiments, the profile will be a list of ratings for a set of attractions in a

specific city. This is data that can be reasonably gather by real systems while at the same time being relatively minimal. We gather this rating information directly from users by asking them to rate a set of attractions in a survey.

We also have to decide what, exactly, an attraction is. For our experiments an attraction consists of a title, short description, and URL that all relate to an attraction. This is very similar to what you would find provided by a general search engine, where the search results consist of a bunch of page titles and descriptions with links that point to the corresponding website.

Besides profiles, the other piece of information we want to know about our travellers is the context that they are searching from. This includes where the user is, for our purposes this is at the granularity of a city and what the time is when the user is searching (this consists of time of the day, day of the week, and season of the year). Again, this is information that real systems would have access to. A system can usually determine a user's location from their IP address to the level of which city they are in. If the user is on a mobile phone there is also GPS data available which can provide a more precise location for the user. Given where the user is searching from the system can determine local time and season. For our experiments we randomly pick a location and time that the user is searching in.

Our research focus is on comparing systems that are able to take in this profile and contextual information about travellers and make helpful suggestions. In order to have these systems developed using a variety of approaches we use TREC. TREC is a conference that has been running since 1992 which provides a venue for research groups from around the world to develop information retrieval systems and compare their systems with systems made by other groups. Each year a handful of specific tasks are decided upon and throughout the year various organizations attempt to develop systems that solve these tasks.

We will go into some of the previous tasks and background of TREC in Chapter 2 however as part of our experiments I developed a TREC task that was approved by the organizers of TREC and then attempted to be solved by several research groups. Charlie Clarke, Jaap Kamps, and Paul Thomas were co-organizers for this track along with me. I lead the efforts for this track and my co-organizers were extremely helpful in providing advice and guidance on how to run the track. The systems that were generated as part of this track will be the systems that we compare in our experiments. In Chapter 3 we will discuss the details of this track further.

The systems in our experiments all take in a profile and a context and for that profile-context pair generate a list of attractions that the system thinks the user will find the most

interesting. Our experiments focus on comparing various systems and determining which one does the best job. As part of the TREC task research groups built systems to create lists of suggestions, downloaded sets of profiles and contexts, ran their systems on each profile-context pair available, and submitted the output of their systems to us.

This research includes creating these profiles and contexts, gathering the output from various systems, and then developing methods of comparing the outputs from the systems. The task of comparing systems is divided up into a judging stage, where individual suggestions in the output are judged for how contextually appropriate they are and how relevant they are to the profile they are. Judging is done both by trained NIST assessors as well as users who we gathered the profiles from. After judging is complete we evaluate the systems. Evaluation is done both by using established metrics and a modification on the Time-Biased Gain framework proposed by Smucker and Clarke [22], which results in a metric catered for this task.

1.2 Organization of the Thesis

The remainder of this thesis is organized as follows:

In chapter 2, we conduct a review of the evaluation of both information systems and recommender systems. We discuss general strategies for how systems can be evaluated and consider a few specific metrics that can be used with system-style evaluation. We also discuss TREC, a venue where many different systems have been evaluated.

In chapter 3, we discuss how profiles and contexts are developed as well as how results from the systems that generate suggestions are gathered. We first look at what a profile in our experiment consists of, how we gather profiles from real users, as well as the format of a context. We then discuss how we gathered suggestions from many different systems and how these suggestions were judged for relevance with respect to both the profiles and the contexts.

In chapter 4, we develop an advanced technique for evaluating contextual suggestion systems that is tailored specifically for evaluating this kind of task. Our metric is based on the Time-Biased Gain framework introduced by Smucker and Clarke [22] which is then customized to meet the needs of our task.

In chapter 5, we evaluate the suggestions given by systems according to several metrics, including our modified Time-Biased Gain metric. We consider the appropriateness and usefulness of each metric as well as the ranking of all the systems.

Finally, in chapter 6, we discuss potential future avenues of research into contextual suggestions systems as well as evaluation metrics for such systems. In particular we consider further modifications to our modified Time-Biased Gain metric as well as how those modifications could be tested.

Chapter 2

Related Work

Information retrieval evaluation can be done using one of two main strategies. The first strategy involves running user studies that let real users try to meet some goal by interacting with the systems directly. The second strategy relies on estimating how well a system would meet a user's needs. In the second style of evaluation, judgments are made about the relevance of the documents returned by systems. These judgments are then used in a calculation that estimates system usefulness. A simple way of calculating this estimate is to score systems by how many relevant documents they return. This score may not be an accurate estimate of system usefulness and we will look at more accurate ways of calculating this estimate later on in this chapter.

User study style testing tests systems in a setting that is closer to how the systems will actually be used than batch-style testing however it is more expensive because real users must spend time interacting with the systems. Batch-style testing allows us to very quickly evaluate a system once we have relevance judgments. For our experiments we will be using batch-style testing. However in this style of testing we still have to decide exactly how a score is calculated, which is based on which model of user behaviour we are using. The simple metric above, i.e., calculating how many relevant documents were returned, is based on the model that users will consider each document that is returned. This would be a time consuming process and is probably not how a user really interacts with a system. Our goal is to find a model which reflects how users really use systems and base our metric on that model. This will ultimately lead to a metric that gives systems which are more useful to users, higher scores.

When deciding on a user model which reflects how users will interact with our systems we should consider the goals of the systems we are evaluating. Systems that return legal

documents and may want to return every relevant document and will be evaluated differently than general information retrieval systems where the user wants to find any relevant document relatively quickly. The systems that we are evaluating makes attraction suggestions to tourists, we will keep this goal in mind when considering evaluation metrics. In this section we will consider some established evaluation metrics that have been used to evaluate other systems. Particularly, we will focus on evaluation metrics that have been used with information retrieval (IR) systems and recommender systems.

The goal of a recommender system is to make item suggestions to users. The system assigns ratings to items that the user has not rated in order to make these item suggestions. Users will have rated other items already and systems will try to make guesses about how users will rate new items based on the ratings they have already given. Commonly systems will use ratings for the items given by other users in order to help them make their prediction. In some systems these predicted ratings are exposed to users directly, or in the form of ranked lists of items, or as the top n recommended items. An example of such a system is the one used by Amazon.com to recommend products [11].

On the other hand, information retrieval systems take in a query, a set of terms, as input and produce a ranked list of documents that are relevant to the particular query. Some documents are filtered out if they are non-relevant, and some documents will be placed higher in the result list, if they are considered more relevant. An example of this kind of system is the one provided by Google Search.

The type of system that we concentrate on in our experiments fall somewhere between traditional IR systems and traditional recommender systems. Our systems don't have an explicit query like traditional IR systems, however we are still searching a large corpus of documents. Like recommender systems, we are trying to suggest items (in our case attractions) that the user would find interesting, however we have no ratings for attractions or even an established list of attractions to draw from when developing our systems. Our systems will be returning documents that represent attractions in some way, for example they could be the official home page of a particular attraction.

In order to compare systems an evaluation metric needs to be decided upon. Our choice in metric should be based on a model that reflects how users will actually interact with our attraction suggestion systems. In this chapter we will review TREC, a conference that has successfully evaluated hundreds of systems using batch-style testing. We will also look at a few of the commonly used metrics used to evaluate recommender and IR systems.

2.1 Text REtrieval Conference (TREC)

A popular venue to test information retrieval system ideas is the Text REtrieval Conference (TREC). When information retrieval researchers develop a concept for a new system an important task they have to do is evaluate their system against other systems. Even when doing system style testing evaluation is not an easy task. The systems all have to be compatible: they have to take in the same input format and product the same output format. Also, in order to be able to evaluate systems fairly they all have to be tested on the same dataset and this dataset has to be judged. Judging is a process that takes a significant amount of resources.

TREC solves these issues by providing a common platform for many different search systems to be evaluated against each other. Since 1992 TREC has developed tasks that researchers from around the world are invited to attempt to solve. The most basic task is adhoc retrieval [6]. For this task systems are provided with topic queries and are asked to search a specific dataset for any documents that are relevant. By coordinating research efforts in information retrieval around a task like this we already have multiple systems with the same input and output specifications and multiple systems using the same dataset.

Because all the systems taking part in this challenge are using the same dataset and queries, we can manually judge the relevancy of documents in this dataset and then compare these manual judgments against system output. TREC regularly invests in having trained assessors judge the relevance of documents in the datasets being used. System output and these judgments can be plugged into evaluation metric calculations to determine which system performed the best, which in turn tells the research community which information retrieval techniques are superior. We will discuss a few options for evaluation metrics in the next two sections.

Since starting, TREC has been the host to many different IR focused tasks. For example, besides adhoc retrieval, there have been tasks on:

- Filtering [18] where, in addition to a query, systems are given, as input, a set of documents that the searcher liked. The focus of this task is to see whether systems can improve results if they know which documents the searcher liked.
- Question answering [23] where systems are asked to directly answer a question, using a short text snippet, that users are looking for instead of simply returning a document that might be relevant to the question.
- Spam filtering [4] where systems are required to give a binary judgments for documents indicating whether they are spam or not.

TREC also has supported many other tracks, for each track a dataset is decided upon, any input, such as queries or training data is provided, then research teams have a few weeks to develop their systems and return results. TREC then organizes the judging process and evaluates the systems, results are then released at the time of the conference.

Each year, TREC supports a different set of tasks, for our experiments we set up a TREC task, which was accepted by the organizers of TREC, in chapter 3 we will discuss how this task was designed, what inputs were used, what outputs were expected, and what datasets were decided upon.

2.2 Evaluating Recommender Systems

The basic task of recommender systems is to try to predict the ratings that users will give to items. One strategy when evaluating these systems is to compare the true rating given by users to the predicted ratings given by the system.

One widely used metric to evaluate recommender systems is *mean absolute error* (MAE), and variations on it [7]. This metric measures the mean of the absolute difference between the actual ratings and the ratings given by the system:

$$\frac{1}{N}\sum_{i=1}^{N}|p_i - r_i|, \tag{2.1}$$

where p_i is the ratings given by the system, r_i is the rating given by the user, and N is the number of ratings. If we replace the absolute value function in the equation above with the square function, we get the *mean squared error*:

$$\frac{1}{N}\sum_{i=1}^{N}(p_i - r_i)^2. \tag{2.2}$$

This metric will magnify errors that the system makes. In this metric, if two systems are compared, one that makes many small errors and one that makes a few large errors, the metric will rank the second system worse than the first. With MAE the two system's performances will be closer together.

A third variation is root mean squared error, which is simply the square root of MSE:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - r_i)^2}. \tag{2.3}$$

Precision and *recall* are two other well known metrics that have been used to evaluate recommender systems. Precision is the number of liked recommendations suggested, divided by the total number of recommendations suggested:

$$\frac{|\text{retrieved} \cap \text{relevant}|}{|\text{retrieved}|}. \tag{2.4}$$

Note that this metric uses binary relevance judgments, i.e., items are grouped into two categories: relevant or non-relevant. The previous three metrics use graded relevance judgments, for example, users could give ratings on a 5-point scale, they could say that an item is one of very good, good, fair, poor, or very poor. For our metrics user ratings are treated as numbers so we could map these ratings to 5 (very good), 4, 3, 2, or 1 (very poor). Depending on our situation we may have more or fewer ratings levels that have been given to items.

When we have ratings given on a 2-point scale (a rating of either good or bad) it is simple to group items into the two categories, items with a good rating will be considered relevant and items with a bad rating will be considered non-relevant. However we can still use precision if we have graded relevance judgments, in order to do this, the items will have to be mapped to relevant or non-relevant. The simplest way to do this is pick a threshold where ratings below the threshold are considered non-relevant and ratings above the threshold are considered relevant. For example, if ratings were on a 5-point scale items with a rating of 4 or 5 could be considered relevant and items with a rating of less than 4 could be considered non-relevant.

Recall is the the number of suggestions made by the system that the user likes, divided by the number of suggestions that the user would like if they had seen them:

$$\frac{|\text{retrieved} \cap \text{relevant}|}{|\text{relevant}|}. \tag{2.5}$$

Here users have rated more items than systems will retrieve because the judging process is separate from the system output and metric calculation step. However even during the judging process users will not see even possible item, there are often many more items than can be reasonably rated by a user. Users will rate some subset of the items available

and every item that is not rated will be considered non-relevant for the purposes of our metrics. So, in the above equation, we usually don't know all the suggestions that the user likes because the user hasn't given a rating for every single item. In this case the number of relevant items is estimated as the number of relevant items *that we have user ratings for*.

Recommender systems often make ordered recommendations to users. In these cases we can make an analogy between evaluation of recommender systems and information retrieval systems. In our analogy recommendations are documents and user ratings are relevance judgments. Using this strategy all the evaluation metrics described in the next section can also be used with recommender systems.

2.3 Evaluating IR Systems

The evaluation of information retrieval systems is traditionally based on document relevance judgments made by human assessors. These judgments are often binary judgments (relevant or non-relevant) but judgments can also be made for greater levels of relevance granularity (graded relevance).

Systems will return ranked (ordered) lists of documents and our goal is to determine how useful this list is which in turn indicates how useful the system is. For example, suppose we have a list of 5 documents that have been assigned the following relevance judgments:

Rank	Document title	Judgment
1	Waterfront Grill	Non-relevant
2	Copeland's	Relevant
3	Cormier's Cajun Catering & Restaurant	Relevant
4	Enoch's Pub & Grill	Relevant
5	Cotton	Non-relevant

Table 2.1: Example relevance judgments.

The output of the system, combined with the relevance judgments, is used to calculate an effective metric, the result of this calculation will then give us an indication of the performance of the system and its usefulness. We want a metric that represents the quality of the system that produced this list of documents. Various metrics have been proposed to represent system utility, we outline a few below.

2.3.1 Precision at rank k

One commonly used metric is *precision at rank k* or P@k which is simply the number of relevant documents on or before rank k divided by k[2]:

$$\frac{\sum_{i=1}^{k} rel(i)}{k}, \tag{2.6}$$

where $rel(i)$ is 1 if the document at rank i has been judged relevant and 0 otherwise. Note that this metric only uses binary relevance judgments. If graded relevance judgments are available then these need to be mapped to binary relevance judgments.

So the P@5, where k is set to 5, of the documents in the above table would be 0.6. The user model corresponding to this metric is a user who looks at the first k documents in the search results before becoming bored and stopping, regardless of how many relevant documents have been seen.

2.3.2 Reciprocal rank

Another possible metric is *reciprocal rank* or RR which is the reciprocal of the rank of the first relevant document[2]:

$$\frac{1}{first(i)}, \tag{2.7}$$

where $first(i)$ is the rank of the first relevant document. So the RR of the documents in the above table would be $\frac{1}{2} = 0.5$. If there are no relevant documents in the list the reciprocal rank will be 0. The user model corresponding to this metric is a user who stops looking through the list of documents once the first relevant document is encountered.

2.3.3 Discounted cumulative gain

Discounted cumulative gain is a metric proposed by Järvelin and Kekäläinen[9], which is widely used in web search contexts. Here we add the relevance judgment for documents up to a certain rank, k, however these judgments are discounted by a certain amount at each rank, commonly the judgments are divided by the log of the rank to provide this discounting:

$$rel(1) + \sum_{i=2}^{k} \frac{rel(i)}{log_2(i)}. \tag{2.8}$$

Again, here $rel(i)$ is the relevance judgments of document i but unlike for P@k these judgments do not need to be restricted to binary judgments. For our example this metric is calculated as follows:

Rank	Document title	Judgment	
1	Waterfront Grill	Non-relevant	0.0000
2	Copeland's	Relevant	1.0000
3	Cormier's Cajun Catering & Restaurant	Relevant	0.6309
4	Enoch's Pub & Grill	Relevant	0.5000
5	Cotton	Non-relevant	0.0000
-	-	Total	2.1309

Table 2.2: DCG calculation.

We calculate the discounted gain for each rank and then sum to get 2.1309. The user model for this metric is somebody who gets less and less likely to read the documents as they progress down the list. Note that here we stopped the calculation at rank k because this is the last rank we have judgements for, typically the calculation will be done up to the last rank judgements are available for.

2.3.4 Rank-biased precision

A fourth metric, proposed by Moffat and Zobel[15], is *rank-biased precision* (RBP). This metric, like DCG, incorporates a discounting factor for each document in the list:

$$(1-p) \cdot \sum_{i=1}^{k} rel(i) \cdot p^{i-1}, \tag{2.9}$$

where p is the probability that the user will continue to the next document in the list, for example it is very likely somebody with $p = 0.99$ will move to the next document in the list however somebody with probability $p = 0.0$ will only read the first document in the list. For RBP the parameter p allows us to adjust how likely the person is to continue.

We can calculate the RBP for our example above, setting $p = 0.95$ and $k = 5$:

Rank	Document title	Judgment	$rel(i) \cdot p^{i-1}$
1	Waterfront Grill	Non-relevant	0.0000
2	Copeland's	Relevant	0.9500
3	Cormier's Cajun Catering & Restaurant	Relevant	0.9025
4	Enoch's Pub & Grill	Relevant	0.8574
5	Cotton	Non-relevant	0.0000
-	-	Total	2.7099

Table 2.3: RBP calculation.

After calculating $rel(i) \cdot p^{i-1}$ for each rank, we take the sum (2.7099) and then multiply by $1 - p$ to get 0.1355. This user model, like the one for DCG, corresponds to somebody who is less and less likely to continue scanning the list as they scan down it.

2.3.5 Expected reciprocal rank

Chapelle et. al., introduced another option for metric choice, *expected reciprocal rank* (ERR)[3]. Like DCG and RBP this metric uses a discount factor for each document in the list, however this factor is based on not only the rank of the document (like the other two) but also on the number of relevant documents that come before the document being considered. ERR can be expressed as:

$$\sum_{i=1}^{k} \frac{1}{i} \prod_{j=1}^{i-1} (1 - rel_n(j)) \cdot rel_n(i), \tag{2.10}$$

where $rel_n(i)$ is proportional to the document relevance judgment:

$$rel_n(i) = \frac{2^{rel(i)} - 1}{2^{max\_rel}}. \tag{2.11}$$

In the previous equation max_rel is the highest judgment a document can get. This is another metric that has the flexibility to use graded relevance, in our example, which uses binary relevance, $max\_rel = 1$, so if the document at rank i is relevant $rel_n(i) = 0.5$, otherwise $rel_n(i) = 0$.

In equation 2.10, the $\frac{1}{i}$ is the discount factor based on position and the product is the discount factor based on the number of relevant documents that are ranked higher in the list. For our example, we can calculate ERR as follows:

Rank	Judgment	$rel_n(i)$	$\prod_{j=1}^{i-1}(1 - rel_n(j))$	$\frac{1}{i}\prod_{j=1}^{i-1}(1 - rel_n(j)) \cdot rel_n(i)$
1	Non-relevant	0.0	0.000	0.0000
2	Relevant	0.5	1.000	0.5000
3	Relevant	0.5	0.500	0.1667
4	Relevant	0.5	0.250	0.0625
5	Non-relevant	0.0	0.125	0.0000
-	-	-	Total	0.7292

Table 2.4: ERR calculation.

We calculate the individual addends and then sum them to get 0.7292. The user model here is slightly modified from DCG and RBP in that the user becomes less and less likely to view a document as they progress down the list but viewing relevant documents will also make the user less likely to continue scanning the list of documents.

2.3.6 Time-biased gain

DCG, RBG, and ERR all have a related user model, a user who scans a list of documents and eventually stops due to some factor, either because they have seen enough relevant documents or have seen too many documents and have gotten bored. Smucker and Clarke introduced a framework that encompassed these metric and has the flexibility to allow other factors to be incorporated[22]. The most general form of this *time-biased gain* (TBG) framework can be expressed as:

$$\frac{1}{N}\int_0^\infty D(t)dG(t). \tag{2.12}$$

The equation assumes that the user is working through a ranked list of retrieval results, reading documents, clicking on links, or performing whatever other actions are appropriate to the retrieval task at hand. The function $G(t)$ represents the cumulative gain, or benefit, received by the user as time passes.

The decay function $D(t)$ indicates the probability that the user continues until time t. This function represents the possibility that the user will stop at some point due to factors such as tiredness or boredom, rather than due to the influence of the results themselves.

N is the maximum amount of gain a user can experience (an optional normalization factor). TBG is simplified as:

$$\frac{1}{N}\sum_{k=1}^{\infty} g_k D(T(k)). \tag{2.13}$$

In this equation, g_k represents the gain realized from the kth item. The function $T(k)$ represents the time it takes the user to reach rank k.

This framework will prove useful when developing our own metric later in this thesis in chapter 4. We will go into more detail of TBG and discuss our extension to it then.

Chapter 3

Experimental Design

The experiments reported in this thesis revolve around the Contextual Suggestion TREC Track, which I was the lead coordinator for, that first ran in 2012 and continued, with the same basic format, in 2013. As the lead coordinator I ran the task as described in this chapter. Besides offering advice and guidance on how to run the task my fellow coordinators sometimes provided help running the task. The most significant contribution from my fellow coordinators was in the building of a subcollection described in section 3.4.

The experiment investigates search techniques for complex information needs that are highly dependent on context and user interests, with a particular focus on "attractions" or "places-of-interest". For example, imagine a traveller visiting Philadelphia, a contextual suggestion system might recommend the Flying Fish Brewery[1] or Elfreths Alley Museum[2]. Our experiment looks at methods of evaluating contextual suggestion systems.

Our strategy is to provide the same input (a profile and a context) to multiple contextual suggestion systems which will then return, as output, a ranked list of attractions the systems thinks is relevant for the profile and context. These ranked lists of attractions will be then judged for relevance. The systems used are developed by participants in the Contextual Suggestion Track. In this chapter we will look at what a profile, a context, and what a list of attractions consist of in detail.

[1] http://flyingfish.com/
[2] http://www.elfrethsalley.org

3.1 Profiles

For our experiments profiles are lists of attractions that a user has given ratings for. A real system might, for example, know which attractions the user likes and dislikes in their home town and use that as input when trying to make suggestions when the user is travelling outside of their home town. The list of attractions rated will be the same for all users, so we first need to decide which attractions users will be rating.

3.1.1 Sample Attractions

Attractions are places that the user can visit. This could be a museum, gallery, restaurant, park, or even a less specific place like a shopping district. For our purposes, the data we have on an attraction is a title, a short description, and a URL that represents a page about the attraction. Two sets of sample attractions were created, one for the 2012 track and one for the 2013 track.

In the 2012 track all sample attractions were gathered from the Toronto, ON region. A list of sample attractions was created from the results of a series of manual searches on commercial search engines. The attractions were chosen based on the quality of information available about them online, while ensuring that there was diversity in the types of attractions put in the list. In order to ensure this diversity, the list of attractions included, for example, art galleries, restaurants, and amusement parks, rather than many attractions all from the same category.

In the 2013 track the sample attractions were gathered from the Philadelphia, PA region. This list was based on attractions returned by contextual suggestion systems in the 2012 track. Again, attractions were chosen for the quality of the descriptions and websites available about them as well as maintaining a diverse list of attractions in an effort to support the preferences of different types of users.

3.1.2 Developing Profiles

Profiles indicate a user's preference for a particular list of attractions, in our experiments this list of attractions is the list of sample attractions described in the previous section. These profiles are then used as input for the contextual suggestion systems. We created an online survey which each user completed, the responses to this survey were used to make up the user's profile. In the survey, users were shown a list of the sample attractions

in the format of a search results page, where they initially saw a list of titles and short descriptions for all the attractions and then had to click through to see the attraction page. These attractions were presented in a random order to each user. Each user was instructed to give two ratings for each attraction. The first rating was the level of interest the user had in going to the attraction after reading the title and description, the second rating was the level of interest the user had in going to the attraction after browsing around on the website for a short amount of time (usually less than a minute).

Figures 3.1, 3.2, 3.3, and 3.4 show samples from the survey used to gather attraction ratings. For each attraction on the first page the user saw and rating the description and then clicked through to the website. In 2012 the rating for the website was gathered directly on the website page. In 2012 we experienced some technical issues with loading pages into HTML IFrames so in 2013 users returned to the original page, once they had viewed the website, to give their website ratings.

In 2012 we recruited users from the University of Waterloo student body to complete our survey, in 2013, in addition to recruiting from the student body, we also used Amazon's Mechanical Turk service to recruit users to complete our survey. In both years the profile consisted of ratings for 50 sample attractions. Approval to run these studies was granted by the Office of Research Ethics at the University of Waterloo.

The level of interest ratings that users gave were on a 3-points scale and a 5-points scale in 2012 and 2013 respectively. In 2012 the three labels for ratings, presented to users were (labels in brackets were shown when the mouse is hovering over the rating button):

0 -1 (Looks boring)
1 0 (Meh, i.e. indifferent)
2 +1 (Looks interesting)

In 2013 the labels were reworded and two extra ones were included:

0 Strongly uninterested
1 Uninterested
2 Neutral
3 Interested
4 Strongly interested

Additionally, for the website, a rating of -2 was given if it didn't load at the time of judgment. In our experiments the time between when a suggestion was make by the system

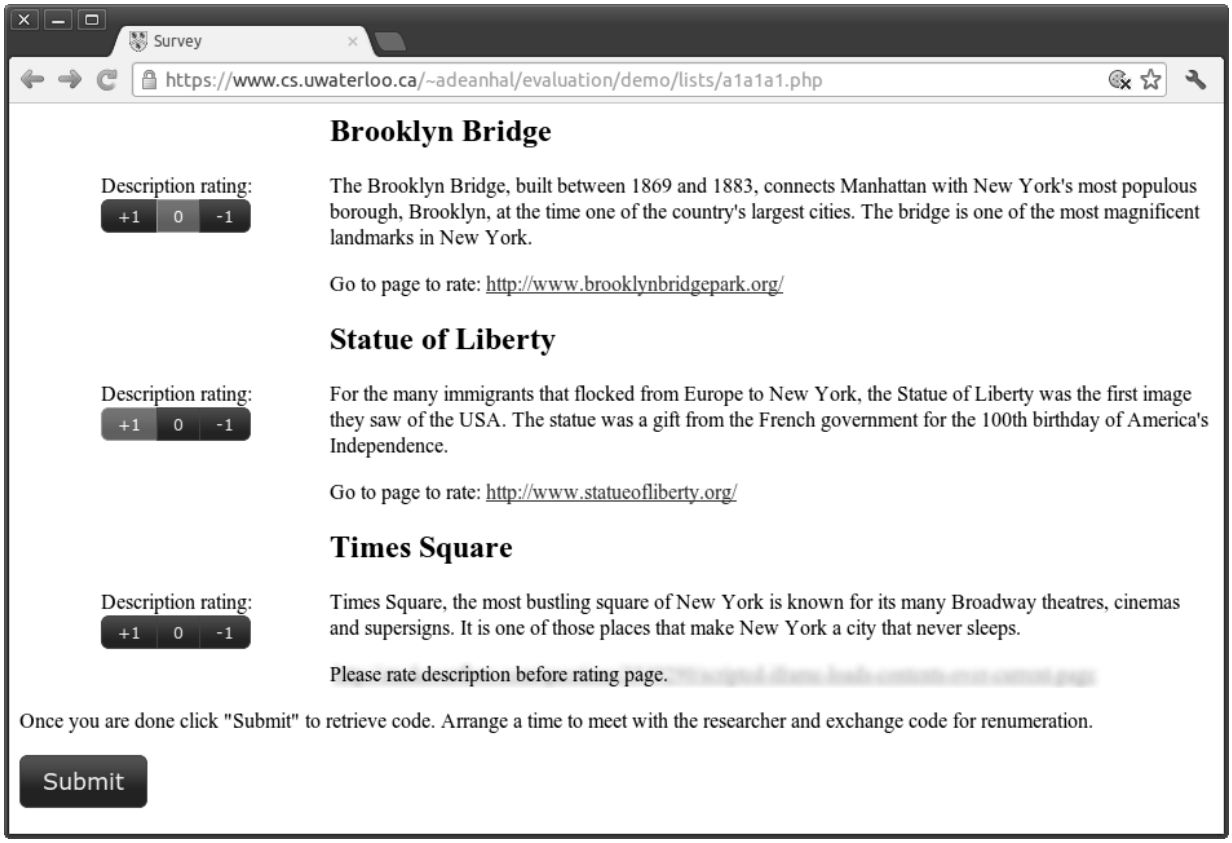


Figure 3.1: Profile gathering and suggestion description judgment interface (2012).

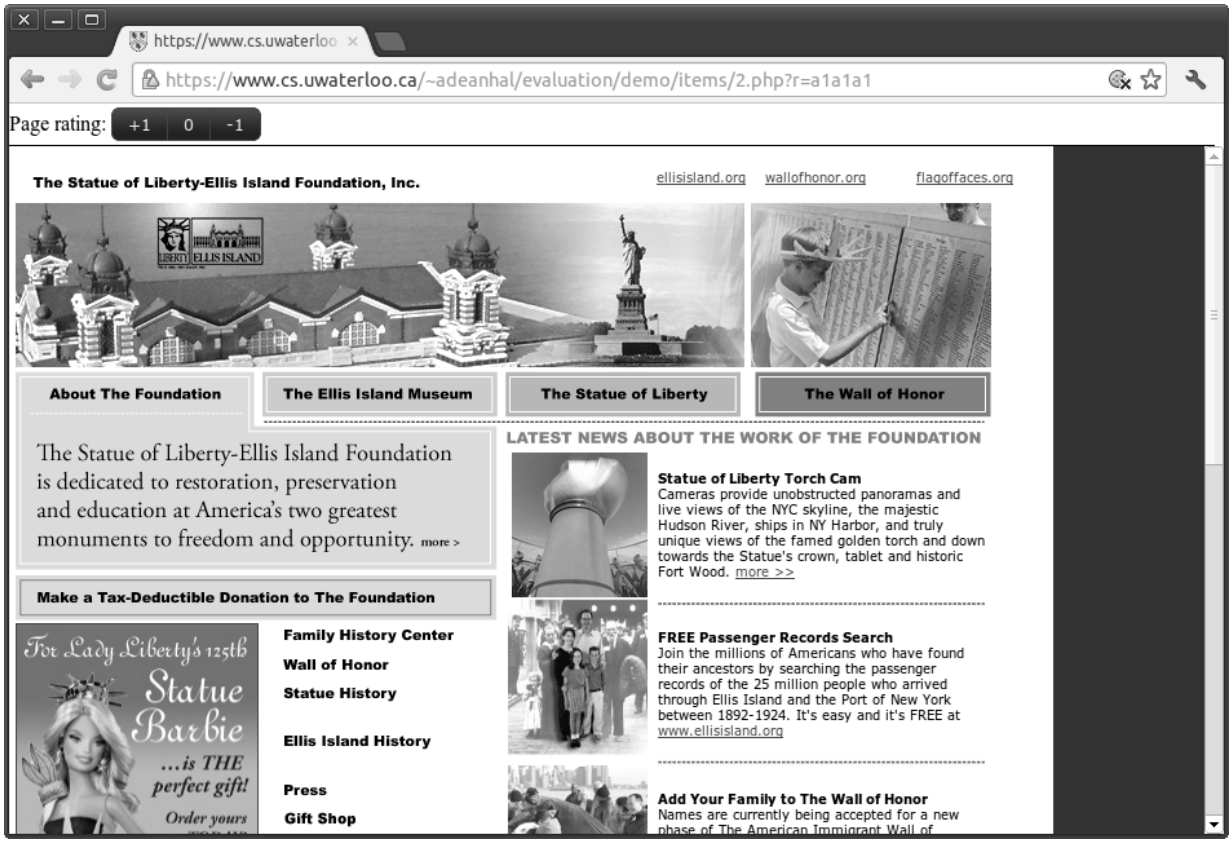


Figure 3.2: Profile gathering and suggestion website judgment interface (2012).

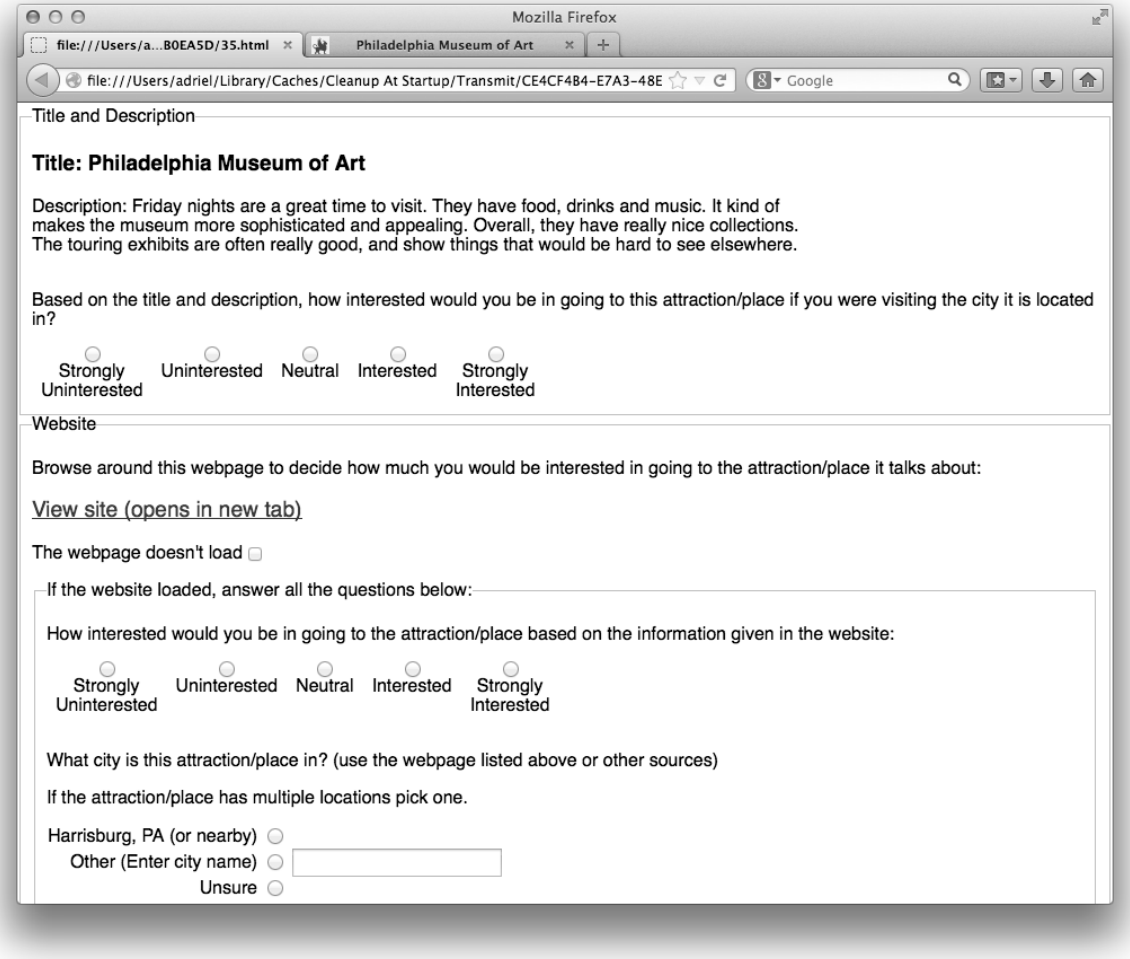


Figure 3.3: Profile gathering and suggestion judgment interface (2013).

Figure 3.4: Suggestion website viewing interface (2013).

and when a user was asked to judge the suggestion was a few weeks. Sometimes the website broke or was taken offline in the delay between suggesting and judging. Other times the website was so slow that the assessor gave up trying to load it and reported a judgment of -2. As we'll see in section 3.4 some of the websites being judged were not live websites on the open web but websites from the ClueWeb12 dataset being mirrored on other servers, occasionally these pages in particular would be offenders in loading slowly.

In the 2012 experiment 34 profiles were created using this method, in the 2013 experiment 562 profiles were created. This large increase in number of profiles was due to the more extensive recruitment done on Mechanical Turk.

Profiles were distributed in two files: sample suggestions and ratings. For 2012, we can see excerpts of the two files in listings 3.1 and 3.2. For 2013, the excerpts are in the listings 3.3 and 3.4.

Listing 3.1: Excerpt of example suggestions file for 2012.

```
<examples>
    <example number="1">
        <title>Fresh on Bloor</title>
        <description>Our vegan menu boasts an array of exotic
           starters, multi-layered salads, filling wraps, high
             protein burgers and our signature Fresh bowls.</
           description>
        <url>http://www.freshrestaurants.ca</url>
    </example>
    ...
</examples>
```

3.1.3 User Quality

In addition to the ratings given by users, timing logs were collected. A timestamp of when a user made a judgment was recorded. For each judgment made, we calculated the difference between each rating's timestamp and the timestamp of the last recorded rating. This gave us an approximation of the amount of time it took to make a judgment, however is inaccurate when users take breaks between judging attractions. The estimate of the time it took to make judgements was used to provide us with an estimate of the quality of the ratings being given.

Listing 3.2: Excerpt of profiles file for 2012.

```
<profiles>
    <profile number="1">
        <example number="1" initial="-1" final="0"/>
        <example number="2" initial="1" final="1"/>
        <example number="3" initial="1" final="1"/>
        ...
    </profile>
    ...
</profiles>
```

Listing 3.3: Excerpt of example suggestions file for 2013.

```
id,title,description,url
51,Elfreths Alley Museum,Elfreths Alley Museum is a reputable
   museum. A lovely little piece of history. Definitely a must
    while visiting Philadelphia... To walk down the oldest
   residential street in the country is just something I think
    everyone should do at least once if in the area! I really
   enjoyed it.,http://www.elfrethsalley.org
...
```

It takes a minimum amount of time to read a description, load a webpage, and read the webpage, if the user is does this very quickly then they have probably not given us high quality ratings. User who didn't meet a certain quality threshold had their profiles marked as bad and were not used as part of the system evaluation metrics.

In the 2013 experiment we expanded this quality assessment to include giving users a question with a known answer for each attraction. As well as collecting timing logs we asked each user, for each attraction, where the attraction was located. All the attractions used to develop profiles were located in Philadelphia, PA, the choices we gave users for this question were:

- Located in Philadelphia, PA
- Located in another city (we asked the user to give the name of the city)
- Unsure where it is located

Listing 3.4: Excerpt of profiles file for 2013.

```
id,attraction_id,description,website
35,51,0,4
35,52,1,4
35,53,3,3
...
```

We also put five attractions that we knew where **not** located in Philadelphia, PA into the survey used to build profiles. For these attractions the same location question was asked with the same options. The answers to these question, in addition to the timing log data, were used to give each user a quality rating. Again users who answered too quickly or didn't get enough of the location questions correct were not included in the metric calculations for system evaluation.

Of the users who completed the initial survey 6% of users got less than 80% of the geographical judgements correct. Additionally, as we will see later, users were invited back to do a second survey, the order that users were invited back in corresponds to how many geographical judgements they got correct, in the end only the top 80% of users were invited back. This number was a combination of the limit of how many users we could invite back (due to resource constraints) and the return rate of the users that we did invite back.

3.2 Forming Contexts

Contexts indicate where a user is physically located and what the time is when the user is searching. The geographical component is at the granularity of the city-level and the user is considered to be looking for any attractions within the city or in nearby cities (short driving distance). The temporal component consists of a part of week (weekday or weekend), a time of day (morning, afternoon, or evening), and a season (spring, summer, fall, or winter).

For the geographical component, in the 2012 track experiment the cities were chosen from the geonames.org list of US cities with a population greater than 100 thousand. The likelihood of being included in the list was proportionate to the population of the city. In the 2013 track experiment the cities were chosen from the wikipedia.org list of primary cities for metropolitan areas in the US[3] (excluding Philadelphia, PA as it was the city used

[3] http://en.wikipedia.org/wiki/List_of_metropolitan_areas_of_the_United_States

for seeding the profiles).

For the temporal component, we selected a part of week, time of day, and season at random in 2012. For the 2013 track experiment we abandoned the temporal component completely leaving only the geographical component to form our contexts. This decision was based on the fact that the temporal component complicated the task without providing any benefit to many of the contextual suggestion systems (many systems ignored the temporal component).

In both years 50 contexts were generated. See listings 3.5 and 3.6 for excerpts of the context files which were distributed to participants in 2012 and 2013 respectively.

Listing 3.5: Excerpt of context file for 2012.

```
<contexts>
    <context number="1">
        <city>New York City</city>
        <state>NY</state>
        <lat>40.71427</lat>
        <long>-74.00597</long>
        <day>weekday</day>
        <time>afternoon</time>
        <season>summer</season>
    </context>
    ...
</contexts>
```

Listing 3.6: Excerpt of context file for 2013.

```
id,city,state,lat,long
51,Springfield,IL,39.80172,-89.64371
52,Cheyenne,WY,41.13998,-104.82025
53,Fargo,ND,46.87719,-96.78980
...
```

3.3 Results

These profiles and contexts were released to research groups, throughout the world, participating in the TREC Contextual Suggestion track. Several groups responded, developed systems and returned the results to us as part of the track.

Each research group was asked to develop one or two systems that took a profile and a context as input and produced a list of up to 50 ranked suggestions (attractions). These attractions are in the same format as the attractions given in the profile, i.e., a title, a short description, and a URL. For each system a list of suggestions was returned for each context-profile pair, this was considered a run. Participants were relatively free in how to implement their systems and what sources they could draw on, for example, systems could make use of data sources such as Yelp or Google Places, however the systems needed to be completely automated.

3.4 Data Collections

In both 2012 and 2013, systems could return URLs from the open web that described the attraction. In 2013 systems were also given the option of returning documents from an offline dataset, ClueWeb12[4].

ClueWeb12 is a dataset created as part of the Lemur Project. It consists of the source of approximately 733 million webpages (documents), which as well as the main text of the page include the HTML, CSS, JavaScript, etc. of the pages, but not the multimedia on the page, for example not the images or videos. Each webpage in the crawl was assigned a unique document id, for example clueweb12-0013wb-88-00134.

The crawl used to capture the source of these pages was conducted in early 2012 and was seeded with about 3 million webpages including about 6 thousand from travel websites. The inclusion of these travel websites makes ClueWeb12 an appropriate dataset for use in this task.

In addition to the full ClueWeb12 dataset two subcollections were created that participants had the option of using for this task, one developed by the same people who developed ClueWeb12 and another developed by a co-organizer of the Contextual Suggestion track. The first subcollection, ClueWeb12 B13 was created by simply uniformly

[4] http://lemurproject.org/clueweb12/

sampling 7% of the full ClueWeb12 dataset, this provides participants who don't have the resources to process the full dataset an option to use.

The second subcollection, ClueWeb12 CS, was developed with the contextual suggestion task in mind. In order to develop this collection a series of queries were issued to a commercial search engine which returned a list of URLs from the open web relevant to the given queries. Queries were of the form "location keywords", where location was the city and state of one of the contexts and keywords was a category of attractions, for example "park".

For each location queries were issued for a variety of categories. The URLs from all of these queries were then filtered on whether the URL had a corresponding document in ClueWeb12 or not (i.e., if it was saved in the crawl done in early 2012). All the webpages that remained were included in the subcollection. This was done for each of the locations in the contexts and the documents in the subcollection were grouped by location.

Participants who choose to use ClueWeb12 rather than the open web to gather suggestions returned ClueWeb12 document ids rather than URLs in their submitted suggestions.

3.5 Example Submission Excerpt

Excerpts of files submitted by participants containing suggestions for 2012 and 2013 are in listings 3.7 and 3.8 for each year respectively.

Listing 3.7: Excerpt of a suggestions file for 2012.

```
<context2012 groupid="waterloo" runid="waterloo12a">
    <suggestion profile="1" context="1" rank="1">
        <title>Manhattan Skyline</title>
        <description>Type: Landmarks/ Points of Interest
           Activities: Helicopter rides</description>
        <url>http://www.tripadvisor.com/Attraction_Review-
           g60763-d267031-Reviews-Manhattan_Skyline-
           New_York_City_New_York.html</url>
    </suggestion>
    ...
</context2012>
```

Listing 3.8: Excerpt of a suggestions file for 2013.

```
groupid,runid,profile,context,rank,title,description,url,docId
UWaterlooCLAC,baselineA,35,73,1,Rensselaer Polytechnic
   Institute,"Rensselaer is America s oldest technological
   research university, offering bachelor s, master s, and
    doctoral degrees in engineering, the sciences, IT and web
   science, architecture, and the humanities",https://www.rpi.
   edu/dept/cct/public/eship/contactus.html,
...
```

3.6 Judging Returned Suggestions

The returned suggestions from each of the runs were judged in two parts. They were first judged according to geographical relevance by both users and trained NIST assessors then according to how well they satisfied the profile by just users. Judgments were done for a subset of the topics (profile-context pairs), which were selected randomly. Most judgments were done up to rank 5 except for some of the topics in 2013 where the trained assessors only judged for geographical appropriateness up to rank 4 due to time constraints. In these cases geographical appropriateness judgements were still available from users.

3.6.1 Profile Relevance

In order to judge the relevance of suggestions with respect to a profile, we conducted a second survey similar to the initial survey used to form profiles. In the second survey, the same users from the first survey were invited to judge suggestions from task participants that were made for their profile. Judgments were split into sessions, where a session consisted of judging every suggestion up to rank 5 from all runs for a certain context for their profile. This meant that a session in 2012 consisted of at most 135 suggestions and in 2013 of at most 170 suggestions as we received 27 runs in 2012 and 34 runs in 2013.

After completing the first session users were invited back to complete a second or sometimes third session (although in 2013 nobody was invited for a third session because we had received an adequate number of judgments). In table 3.1 we see how many users participated in the judging process, how many sessions judging was completed in, and how many suggestions in total were judged. Note that not every user who contributed to

	# Profiles	# Users Judged	# Sessions	# Suggestions
2012	34	19	44	5940
2013	562	127	213	35988

Table 3.1: Number of suggestion judged with respect to profile relevance.

creating a profile was part of the judging process because some users did not respond to invitations to return for a second survey and others were disqualified.

As we did when developing profiles, all suggestions were presented to users in a random order while judging. Again two judgments were requested, one for the description and one for the website. Users were asked to score suggestions with the same rating levels as before and we continued to collect timing data.

3.6.2 Context Relevance

In order to judge the relevance of suggestion with respect to context, assessors at NIST reviewed some of the suggestions. Assessors visited the attraction's website and looked for hours of operation as well as the location of the attraction. Each judged suggestion was given a geographical and temporal judgment:

-2 Could not load
0 Not appropriate
1 Marginally appropriate
2 Appropriate

In 2013, the temporal judgment was dropped because, as we saw in section 3.2 there was no longer a temporal component in the context. Also users, in addition to NIST assessors, were asked to give judgments on how contextually appropriate a suggestion was, although users were not given the opportunity to assign a judgment of "marginally appropriate" (1). Of the suggestions judged for context by both NIST assessors and users there was an agreement of judgments of 77% if judgments of "marginally appropriate" (1) and "appropriate" (2) are considered the same.

3.7 Evaluation of Runs: P@5 and MRR

In order to evaluate the contextual suggestion systems we used the metrics precision at rank 5 (P@5) and mean reciprocal rank (MRR). For each attraction we calculated whether or not it was relevant using binary relevance based on various combinations of description, website, geographical, and temporal judgments. For example, an attraction might be considered relevant if it is geographically appropriate (or marginally appropriate) and both the description and website look interesting (or strongly interesting). This definition of relevance was the main one used and the results in Chapter 5 are based off of this definition. Table 3.2 shows whether 5 example suggestions are relevant or not for this definition.

Rank	Document title	Description	Website	Geographical	Relevant
1	Waterfront Grill	Interested	Strongly Interested	Appropriate	Relevant
2	Copeland's	Interested	Strongly Interested	Appropriate	Relevant
3	Cormier's Cajun Catering & Restaurant	Interested	Uninterested	Not Appropriate	Non-relevant
4	Enoch's Pub & Grill	Uninterested	Neutral	Appropriate	Non-relevant
5	Cotton	Neutral	Interested	Appropriate	Non-relevant

Table 3.2: Binary relevance based on description, website, and geographical appropriateness. Dark cells indicate what made the suggestion non-relevant.

This definition allows us to consider suggestions relevant if they are relevant according to all categories, however, we might be interested in only looking at specific aspects of suggestions. For example, if we are only interested in which systems made the best suggestions regardless of geographical appropriateness we could ignore the geographical judgments and only consider whether the description and website looks interesting when deciding if an attraction is relevant.

The P@5 scores are simply how many attractions are relevant in the first five attractions in the ranked list of suggestions divided by 5. MRR scores are 1 divided by the rank of the first relevant attraction.

In Chapter 5 we will discuss how the systems in our experiments performed when compared with these two metrics as well as with a modified version of TBG which we will discuss next in Chapter 4.

Chapter 4

Evaluation Metric Design

P@5 and MRR are metrics that were both designed for evaluation of generic information retrieval tasks. There are many other metrics also designed to evaluate similar tasks, such as discounted cumulative gain [9], rank-biased precision [15], expected reciprocal rank [3], and many others. While the contextual suggestion task is an information retrieval task it might be possible to develop an evaluation metric that is suited better for this task in particular.

4.1 Motivation

Neither P@5 nor MRR are ideally suited to contextual suggestion. P@5 assumes that the user always views exactly 5 suggestions never more, never less. MRR assumes that the user stops at the first suggestion they like. Both measures ignore the impact of descriptions and negative suggestions, which may cause the user to abandon the results. In this chapter, we describe a metric that takes into account these factors and is tailored more closely to our task.

4.2 Time-biased Gain

To accommodate the limitations of traditional information retrieval measures Smucker and Clarke [22] introduced the TBG framework. We turn to this framework to help us create

a measure that accommodates the custom aspects of our task. This framework uses time-based calibration to account for the impact of user choices and actions. We specialized this framework to create a version of TBG specifically geared to our contextual suggestion task[5], below we describe TBG and the modifications we made to it.

As seen earlier, a general form of TBG may be written as the Riemann-Stieltjes integral:

$$\int_0^\infty D(t)dG(t). \tag{4.1}$$

In this equation, we have removed the optional normalization factor from equation 2.12.

The function $G(t)$ represents the cumulative gain, or benefit, received by the user as time passes.

The decay function $D(t)$ indicates the probability that the user continues until time t. This function represents the possibility that the user will stop at some point due to factors such as tiredness or boredom, rather than due to the influence of the results themselves.

Based on an analysis of a log from a commercial search engine, Smucker and Clarke suggest an exponential decay function with a half-life of $H = 224$ seconds. In the absence of other information, we adopt the same decay function for our version of TBG:

$$D(x) = \exp(x * -\frac{\log(2)}{H}). \tag{4.2}$$

When gain is realized as a step function, e.g., increasing by a fixed amount when the user views a suggestion they like, equation 4.1 may be re-expressed as a sum over documents, suggestions, or other discrete retrieval items:

$$\sum_{k=1}^{\infty} g_k D(T(k)). \tag{4.3}$$

In this equation, g_k represents the gain realized from the kth item. In the case of contextual suggestion, we measure gain as the number of suggested webpages the user views and likes. The function $T(k)$ represents the time it takes the user to reach rank k. The decay function is applied to this time to determine the portion of users who reach rank k.

This equation assumes that the user is working through a ranked list of retrieval results, reading documents, clicking on links, or in our case considering suggestions for attractions.

We provide estimates for g_k and $T(k)$ below.

4.2.1 Gain

To estimate g_k, we borrow an idea from the cascade model of browsing behaviour of search results [3, 26]. Under the cascade model, the gain realized at rank k depends on the relevance of documents appearing at ranks 1 to $k-1$. As more relevant documents are seen by the user, the more likely they are to stop browsing, since their information need may be satisfied.

For contextual suggestion, we use a cascade-like model to account for disliked suggestions. As more disliked suggestions are seen, the more likely the user stops browsing. In equation 4.7 as more disliked suggestions are seen the potential gain from further suggestions goes down. The amount that this goes down is based on the parameter θ.

We define a function indicating if the user likes the suggestion at rank k as follows:

$$A(k) = \begin{cases} 1, & \text{if the user likes or is neutral about the description at rank } k \text{ and also likes the (geotemporally appropriate) webpage at rank } k \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Thus, the user likes a suggestion only if they don't dislike the description, and, after clicking through to the webpage, they like it. Geotemporal appropriateness is considered only at the webpage level: the user never likes a webpage unless it is geotemporally appropriate.

We define a function indicating if the user dislikes the suggestion at rank k as follows:

$$Z(k) = \begin{cases} 1, & \text{if the user dislikes the description at rank } k \text{ or if ther user likes or is neutral about the description but dislikes the webpage} \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

Thus, the user dislikes a suggestion if they dislike the description, but also can dislike a suggestion if they don't dislike the description, but after clicking through they end up disliking the website.

Using this we can calculation how many suggestions the user dislikes after reaching rank i in the list of suggestions:

$$h_i = \sum_{j=1}^{i} Z(j). \tag{4.6}$$

We now define g_k in terms of the user's likes and dislikes as they browse a ranked list of suggestions, as:

$$g_k = A(k)(1-\theta)^{h_{k-1}}. \tag{4.7}$$

If the user views and likes the suggestion at rank k, they receive a gain of 1, but this gain is attenuated according to the number of disliked suggestions seen at ranks 1 to $k-1$. The parameter θ ($0 < \theta < 1$) indicates the probability that the user will stop browsing after viewing a disliked suggestion. In the absence of other information, we adopt a value of $\theta = 0.5$. Note that under this model neutral documents have no impact nor any gain.

Note that this gain function is based on our assumptions about how users read suggestion lists. The users in our experiments were focus on judging while looking at suggestions are were actually viewing every suggestion description and website rather than only clicking on websites that interested them for some reason. An area of future work would be to run another experiment where users are given instructions to simply read a suggestion list with the goal of finding interesting attractions rather than rating each and every suggestion. This experiment would give us a better idea of how users actually browser suggestion lists.

4.2.2 Time

The time to reach rank k, $T(k)$, on this interface, may be estimated from actual user behaviour captured during the judgment process. Using timing logs from potential travellers, we compute the mean time it takes for users to read a description, T_D, and the mean time it takes users to examine a webpage, T_W. Often the suggested webpage is the front page of a large site describing the suggestion and may contain Flash, banners, etc., potential travellers may have looked at only the webpage suggested by the system, or may have clicked through to additional, linked, pages. Examination of these additional pages is included in the times to examine the suggested webpage.

In estimating the time taken to examine a document, Smucker and Clarke [22] consider the document's length. Since users are allowed to click on links and view webpages other than the webpage associated with the suggestions, we do not know the length of the website. Therefore we do not take it into account the website length when calculating the average

time to judge websites. In addition, since descriptions are limited, by the task, to 512 characters, and are generally close to that length, the length of descriptions are not taken into account when calculating the average time to judge descriptions.

As part of the judging process, users clicked through to every website regardless of whether or not they liked the description. In building our model, we assume real users would exhibit different behaviour, only clicking through to pages with a description they like. Under our model, users read every description, and if they like a description they will click through to the website and examine it. Thus, the time to reach rank k is expressed as:

$$T(k) = \sum_{j=1}^{k-1} T_D + l_j T_W. \tag{4.8}$$

where l_j is 1 if the user likes the description at rank j, and therefore examines the webpage, otherwise it is 0.

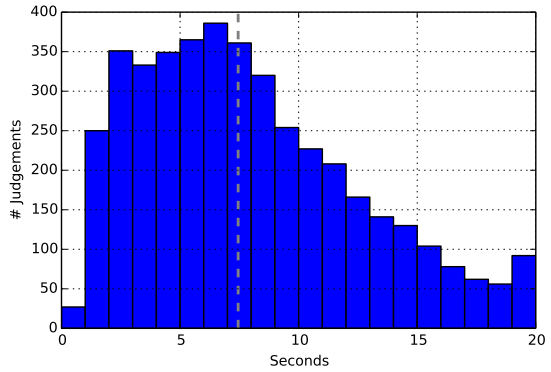
The time that judgements were made was recorded. In order to estimate how long it took users to make judgements we subtract the timestamp of a judgment from the timestamp of the previously judged item. This gives us the length of time it took users to make each description and website judgements. In order to estimate how long it takes a user in general we take the mean of all the times users took to make judgements. However before calculating this mean we removed the slowest 10% of judgements.

When making judgements users sometimes took a break from judging and did something else unrelated to judging before returning to their task. This meant that it looked like users took a very long time to make judgements. Removing the lowest 10% was done in order to eliminate these cases. 10% was chosen in order to gives us a reasonable standard deviation. For example, in 2012, before removing these judgements the standard deviation is approximately 6500 and after removing the slowest judgements the standard deviation is 5.

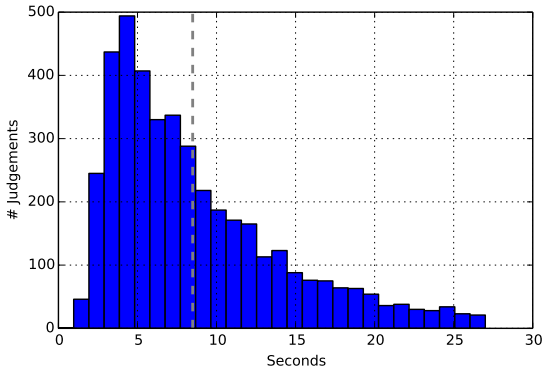
We used the amount of time it took users to make judgements as the amount of time it took users to read descriptions and websites. For 2012, this calculation gives us $T_D = 7.45\,\text{sec}$ for the amount of time it takes users to read descriptions and $T_W = 8.49\,\text{sec}$ for the amount of time it takes users to read website judgements. One slightly surprising observation is that the amount of time it takes users to read websites isn't that that much longer than the amount of time it takes to read a description. The instructions given to users was to read the description and then visit the website for a short period of time. Our guess is that because users were visiting every website in the list of suggestions for the

purposes of judging they spent less time on each suggestion than they would have if they only visited websites with interesting descriptions.

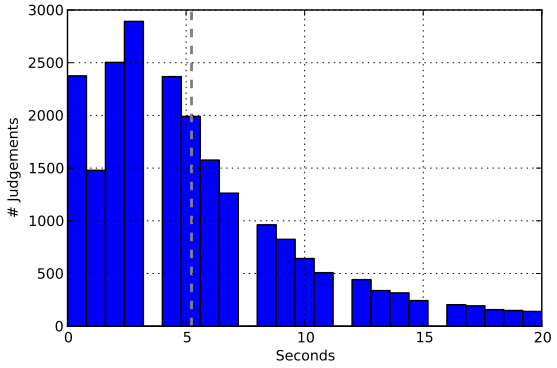
See figure 4.1 for how long it took to make judgments on descriptions and websites. Note that our measures for judgement time were taken in seconds and in some of these histograms there are more bins than seconds so some of the bins are empty. These graphs show the general trend for how long judgements take and where the mean judgement time lies.



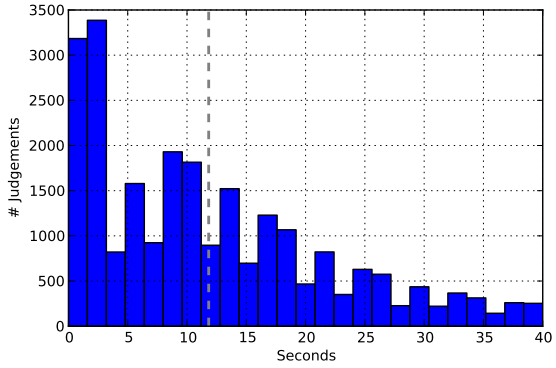
(a) Description judgment time for 2012, mean (marked) is 7.45sec.



(b) Website judgment time for 2012, mean (marked) is 8.49sec.



(c) Description judgment time for 2013, mean (marked) is 5.23sec.



(d) Website judgment time for 2013, mean (marked) is 11.83sec.

Figure 4.1: Judgments times

Chapter 5

Results

There are two sets of data included in our results. In our 2012 experiment there were 27 contextual suggestions systems (or runs) being compared. In 2013 participation increased slightly with 34 runs submitted. We used three metrics for evaluating runs: Precision at rank 5 (P@5), Mean Reciprocal Rank (MRR) and Time-Biased Gain (TBG) with variations on on P@5 and MRR. Table 5.1 shows the rankings of the 2012 runs and table 5.2 shows the rankings of the 2013 runs.

5.1 P@5 and MRR

P@5 is calculated by taking the sum of the number of relevant documents in the first 5 suggestion in our ranked lists of suggestion and dividing by 5. MRR is calculated by taking 1 over the rank of the first relevant suggestion (or 0 if there are no relevant suggestions). For both these metrics we have to determine when a suggestions is considered relevant.

There are three judgments given for each suggestion: a geographical relevance judgment, a description judgment, and a document judgment. For our 2012 experiment there was also a temporal relevance judgment. For the primary scores calculated for TREC in 2012 suggestion were considered relevant if they were geographically appropriate, and both the description and document was interesting. So all three had to have a score of 2 to be considered appropriate (in 2012 all judgments were on a 3-point scale).

In 2013, when description and document judgments were changed to a 5-point scale, a suggestion is considered relevant if it is at least somewhat geographically appropriate (a score of 1 or more), and both the description and document are at least interesting (a

score of 3 or more). For the geographical aspect the change for required a score of 2 to considering 1 good enough was made because the judgments made by users were only on a 2-point scale (although the NIST judgments were still on a 3-point scale).

In addition to these primary relevance definitions (which are the ones used in table 5.1 and 5.2) we also could have used different definitions of relevance which concentrated on and isolated specific judgments.

The relevance definitions above were chosen to represent systems that performed well overall. Other options for relevance definitions could have been used if we want to find out what system performed well in one specific aspect of the task. For example, if we wanted to find out what systems were able to correctly identify location we could have treated all suggestions that were geographically appropriate as relevant, ignoring the description and document judgments. On the other hand, we could have treated all suggestions where both the description and document were interesting as relevant, and ignored the geographical judgment, if we wanted to see what system was best of predicting interesting things regardless of context. Other options for the definition of relevance are also possible if we are interested in considering other aspects of the systems.

5.2 TBG Results

In addition to P@5 and MRR we also consider TBG scores which are calculated as described in chapter 3. The only difference between the 2012 and 2013 experiments for these number is the threshold for acceptability for the description and document judgments. In 2012 it was 2 and in 2013 it was 3, again this change was a consequence of the change from a 3-point to a 5-point scale for judgments.

The TBG system scores for 2012 and 2013 can be seen in igures 5.1 and 5.2 respectively.

5.3 Baseline runs

In addition to using metrics to compare runs against each other we also implemented very simple basic systems which we compared to the other runs. Four baseline runs were created in 2012, two as part of our experiments and two by co-organizers of the Contextual Suggestion track. In 2013 only 2 baselines were created.

5.3.1 2012 Baseline runs

In 2012 two baselines were developed one with no personalization and one with very simple personalization. For the first one, waterloo12a each ranked list of suggestions returned consisted of the top 50 attractions, according to the given rating for attractions, returned for each city from a commercial service (http://tripadvisor.com). For waterloo12b the same service was used except that the service's search tool was also to search for attractions similar to the ones that the particular user gave a high rating for.

This was done by assigning a few terms to each attraction in the profile and searching for each of those terms for each city. If the user liked the description and document in the profile then we used the results from those searches when forming a personalized ranked list of attractions. The result lists were merged and ranked by using the rating that the site gave to each attraction.

Note that for our two baselines, most runs perform better however there are several that perform worse. Also the unpersonalized baseline performs better than the personalized one. However this is likely because of the simplicity of the personalization technique as several of the runs that perform better than the baselines utilize personalization.

As a note, the two baselines produced outside of our experiment for 2012, baselineA and baselineB used another commercial service (Google Places) to search, return lists of attractions, without personalization. BaselineA is what Google Places gives with some minor filtering and baselineB is the same as baselineA except with attractions limited to pubs, restaurants, and cafes.

5.3.2 2013 Baseline runs

In 2013 we created two baselines runs focusing on highlighting the differences between ClueWeb12 and the open-web as choices for where to gather attractions from. Again, a commercial service was used, this time Google Places (which ranks attraction), in order to gather and rank the list of attractions. No personalization is done in either of these runs. In baselineA the top 50 attractions that Google Places returns for a city is used as the ranked list of attractions with open web urls. For the description, a Google Places provided description, review, or a blurb from the meta-description tag on the website is used. In baselineB the same strategy is used except that the urls are mapped to ClueWeb12 documents. Attractions without corresponding ClueWeb12 documents are filtered out.

Note that most open web runs perform better than baselineA however all ClueWeb12 runs perform worse than baselineB. Also note that baselineB performs slightly better than

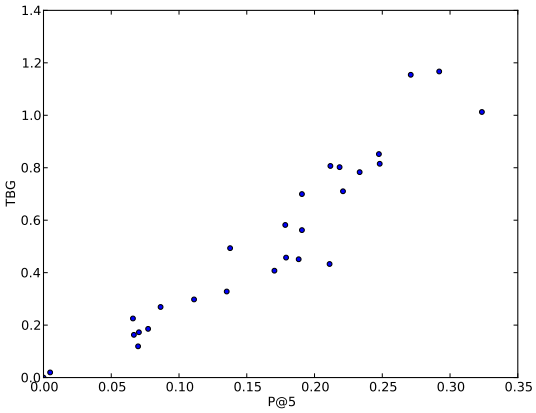
baselineA.

Run	P@5 Score	MRR Score	TBG Score	P@5 Rank	MRR Rank	TBG Rank
iritSplit3CPv1	0.3235	0.4675	1.0126	1	1 (-)	3 (Down 2)
guinit	0.2920	0.4492	1.1670	2	3 (Down 1)	1 (Up 1)
gufinal	0.2710	0.4514	1.1544	3	2 (Up 1)	2 (Up 1)
UDInfoCSTc	0.2481	0.4195	0.8151	4	4 (-)	5 (Down 1)
PRISabc	0.2475	0.4086	0.8521	5	5 (-)	4 (Up 1)
hplcranking	0.2333	0.3868	0.7832	6	7 (Down 1)	8 (Down 2)
UDInfoCSTdc	0.2210	0.3668	0.7103	7	8 (Down 1)	9 (Down 2)
run02K	0.2185	0.3643	0.8022	8	9 (Down 1)	7 (Up 1)
hplcrating	0.2117	0.4037	0.8068	9	6 (Up 3)	6 (Up 3)
udelp	0.2111	0.3118	0.4330	10	14 (Down 4)	16 (Down 6)
ICTCONTEXTRUN2	0.1907	0.3010	0.5622	12	15 (Down 3)	12 (-)
run01TI	0.1907	0.3307	0.6996	11	13 (Down 2)	10 (Up 1)
udelnp	0.1883	0.3395	0.4511	13	11 (Up 2)	15 (Down 2)
iritSplit3CPv2	0.1790	0.3377	0.4574	14	12 (Up 2)	14 (-)
baselineA	0.1784	0.2993	0.5818	15	16 (Down 1)	11 (Up 4)
baselineB	0.1704	0.3504	0.4075	16	10 (Up 6)	17 (Down 1)
waterloo12a	0.1377	0.2130	0.4934	17	18 (Down 1)	13 (Up 4)
UAmsCS12wtSUM	0.1352	0.1727	0.3281	18	19 (Down 1)	18 (-)
ICTCONTEXTRUN1	0.1111	0.2346	0.2979	19	17 (Up 2)	19 (-)
waterloo12b	0.0864	0.1404	0.2691	20	20 (-)	20 (-)
csiroth	0.0772	0.1237	0.1857	21	22 (Down 1)	22 (Down 1)
UAmsCS12wtSUMb	0.0704	0.1058	0.1728	22	24 (Down 2)	23 (Down 1)
csiroht	0.0698	0.1281	0.1191	23	21 (Up 2)	25 (Down 2)
FASILKOMUI02	0.0667	0.1163	0.1629	24	23 (Up 1)	24 (-)
FASILKOMUI01	0.0660	0.0800	0.2253	25	25 (-)	21 (Up 4)
watcs12a	0.0049	0.0062	0.0196	26	26 (-)	26 (-)
watcs12b	0.0000	0.0000	0.0000	27	27 (-)	27 (-)

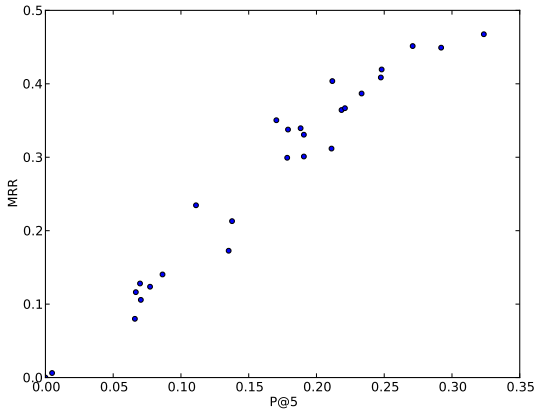
Table 5.1: P@5, TBG, and MRR rankings for all 2012 runs.

Run	P@5 Rank	P@5 Score	TBG Rank	TBG Score	MRR Rank	MRR Score
UDInfoCS1	1	0.5094	1 (-)	2.4474	1 (-)	0.6320
UDInfoCS2	2	0.4969	2 (-)	2.4310	2 (-)	0.6300
simpleScore	3	0.4332	4 (Down 1)	1.8374	4 (Down 1)	0.5871
complexScore	4	0.4152	5 (Down 1)	1.8226	6 (Down 2)	0.5777
DuTH_B	5	0.4090	3 (Up 2)	1.8508	3 (Up 2)	0.5955
1	6	0.3857	8 (Down 2)	1.5329	7 (Down 1)	0.5588
2	7	0.3731	7 (-)	1.5843	5 (Up 2)	0.5785
udel_run_D	8	0.3659	9 (Down 1)	1.5243	8 (-)	0.5544
isirun	9	0.3650	6 (Up 3)	1.6278	9 (-)	0.5165
udel_run_SD	10	0.3354	16 (Down 6)	1.2882	10 (-)	0.5061
york13cr2	11	0.3309	12 (Down 1)	1.3483	15 (Down 4)	0.4637
DuTH_A	12	0.3283	14 (Down 2)	1.3109	12 (-)	0.4836
york13cr1	13	0.3274	15 (Down 2)	1.2970	14 (Down 1)	0.4743
UAmsTF30WU	14	0.3121	17 (Down 3)	1.1905	13 (Up 1)	0.4803
IRIT.OpenWeb	15	0.3112	10 (Up 5)	1.4638	11 (Up 4)	0.4915
CIRG_IRDISCOA	16	0.3013	18 (Down 2)	1.1681	16 (-)	0.4567
CIRG_IRDISCOB	17	0.2906	20 (Down 3)	1.1183	19 (Down 2)	0.4212
uncsils_param	18	0.2780	13 (Up 5)	1.3115	18 (-)	0.4271
uogTrCFP	19	0.2753	11 (Up 8)	1.3568	17 (Up 2)	0.4327
ming_1	20	0.2601	22 (Down 2)	1.0495	22 (Down 2)	0.3816
uncsils_base	21	0.2565	19 (Up 2)	1.1374	20 (Up 1)	0.4136
ming_2	22	0.2493	23 (Down 1)	0.9673	23 (Down 1)	0.3473
uogTrCFX	23	0.2332	21 (Up 2)	1.0894	21 (Up 2)	0.4022
run01	24	0.1650	24 (-)	0.7359	24 (-)	0.2994
baselineB	25	0.1417	26 (Down 1)	0.4797	25 (-)	0.2452
baselineA	26	0.1372	25 (Up 1)	0.5234	26 (-)	0.2316
BOW_V17	27	0.1022	28 (Down 1)	0.3389	28 (Down 1)	0.1877
BOW_V18	28	0.1004	27 (Up 1)	0.3514	27 (Up 1)	0.1971
IRIT.ClueWeb	29	0.0798	29 (-)	0.3279	29 (-)	0.1346
RUN1	30	0.0628	30 (-)	0.2069	30 (-)	0.1265
RUN2	31	0.0565	31 (-)	0.2020	31 (-)	0.1223
csui02	32	0.0565	32 (-)	0.1785	32 (-)	0.1200
csui01	33	0.0565	33 (-)	0.1765	33 (-)	0.1016
IBCosTop1	34	0.0448	34 (-)	0.1029	34 (-)	0.0569

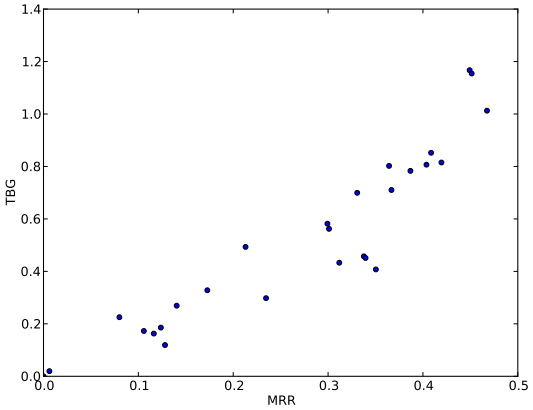
Table 5.2: P@5, TBG, and MRR rankings for all 2013 runs. Bold indicates a ClueWeb12 run.



(a) P@5 vs TBG $\tau = 0.8502$

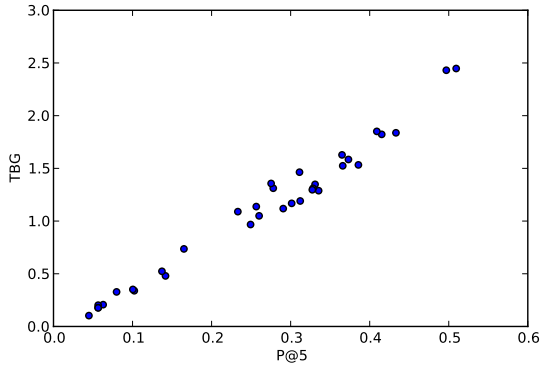


(b) P@5 vs MRR $\tau = 0.8730$

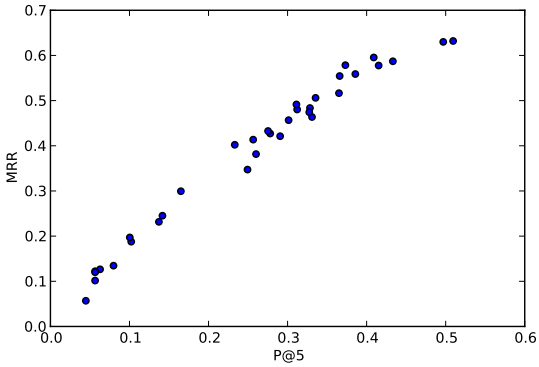


(c) MRR vs TBG $\tau = 0.7949$

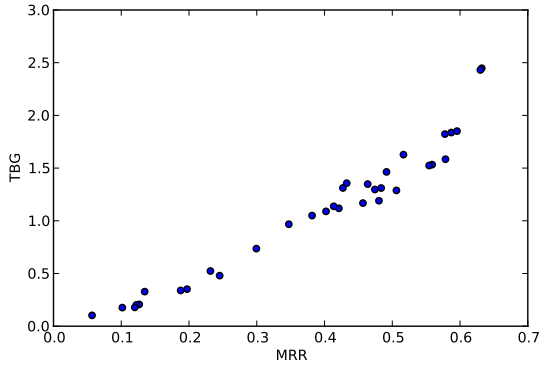
Figure 5.1: Comparisons between P@5, MRR, and TBG for 2012.



(a) P@5 vs TBG $\tau = 0.8160$



(b) P@5 vs MRR $\tau = 0.8959$



(c) MRR vs TBG $\tau = 0.8632$

Figure 5.2: Comparisons between P@5, MRR, and TBG for 2013.

5.4 System and Metric Comparisons

Tables 5.1 and 5.2 show the system scores for the 2012 and 2013 experiments. There are a few things to take note of in these tables. Firstly, we can compare the systems against the default results provided by commercial services in our baselines systems. For 2012, the baseline systems (baselineA, baslineB, waterloo12a, and waterloo12b) all performed very similarly to each other are were approximately the median scoring systems.

Systems that performed worse than the baselines didn't improve on existing commercial services already available to users, however many systems performed better than the baselines and potentially improve on existing commercial services.

For 2013 we can again note the position of the baseline systems (baselineA and baselineB), however this time we note that many of the systems improved on the baselines. Remember, for 2013 a second dataset could be used by participants (ClueWeb12 rather than the open web). Runs that choose to use this dataset are marked in bold. Although these runs scored worse than the open web runs we cannot make any claims about the performance between the ClueWeb12 and open web runs. Only runs that operate on the same dataset can be compared to each other.

The second thing to note in these tables is the amount of shifting of ranks is done when ordering runs by the three different metrics (P@5, MRR, and TBG). These tables are ordered by P@5 rank and for the TBG rank and MRR rank columns the move in position up or down from the P@5 rank has been noted. Also the darker the colour of the cell in these columns the greater the movement in rank.

For 2012, we can note that there is some movement in rank when changing the ordering metric used but most systems stay at a similar rank, the greatest change is a move by 6 rank positions. We are especially interested in the movement near the top of the ranked list of systems because ultimately we are most interested in the top performing systems. Here we can see some movement but the top 3 systems remain the top 3 systems (in a different order) regardless of the metric used.

For 2013, a similar situation can be seen, with slightly larger shifts in the middle of the list, the largest being a shift of 8 positions. Again, in the top scoring systems there is little movement, the top two systems remain exactly the same regardless of metric and the third best scoring system according to P@5 moves down by 1 position according to the other two metrics.

Figures 5.1 and 5.2 show the correlation between the scores given by the three metrics more clearly than the tables do. Again, by looking at these graphs the same conclusion

can be drawn: there is little shift in the ranks given to systems by changing the metric used. The Kendall's tau coefficient is high (above 0.8) in all cases except when comparing MRR and TBG for the 2012 experiments where it is slightly lower at 0.79.

The main conclusions that can be seen from these results is that systems can potentially improve on existing commercial services and that there is not much difference in system position ranking with different metric choices. This doesn't help much when trying to choose a metric but, as discussed briefly in Chapter 6, the choice of which metric to use is made in part by how closely the user model that the metric follows the behaviour of actual users.

5.5 Participant Approaches

14 teams participated in the TREC experiments in 2012 and 19 teams participated in 2013. The teams are listed in tables 5.3 and 5.4 for 2012 and 2013 respectively.

Participants used various different strategies to implement their systems, however there were common themes among these system. Described below are the general strategies that participants used.

5.5.1 2012 Participant Approaches

Although each participant's system in the 2012 experiment was different, every system followed a similar framework. For each profile-context pair systems were required to output a list of up to 50 ranked suggestions. In order to do this systems first gathered a set of candidate suggestions. These suggestions consisted of, at least, a URL but often had other information associated with them, for example, ratings, descriptions, titles, etc. These candidate suggestions were then filtered and sorted based on the profile and context. Finally description snippets were generated using strategies from returning a commercial search engine's snippet for the URL to composing sentences from multiple sources.

For the first step of gathering candidate suggestions systems crawled general search engines such as Google or Bing, travel listings sites such as TripAdvisor[1] or Yelp[2], or queried APIs such as the Google Places API [16, 12, 14, 13, 19, 10, 24, 8]. A common strategy was to query or crawl pages for specific locations so that sets of candidate suggestions were

[1] http://tripadvisor.com
[2] http://www.yelp.com

developed on a per-context basis [16, 12]. Some systems also searched for specific keywords that appeared in the example suggestions to further direct their crawl [10].

Another strategy used by systems was to search by venue category, for example "park", which allowed them to group candidate suggestion by category in addition to location [25].

Some candidates did no filtering by either category, keyword, or location and simply gathered a list of candidate suggestions. For filtering based on location, these systems then gathered the address and geographical coordinates for candidate suggestions from the service used directly or from the URL of the candidate suggestion. A comparison between the location of the candidate and the location in the context was then done and candidates that were too far from the location in the context were filtered out [13].

Some systems used more than one service to gather candidate suggestions. If the candidate suggestion appeared in more than one service the systems had to detect and merge or remove duplicated suggestions. Strategies used to do this were by matching by name, matching by URL, matching by proximity, and some combination of these [14, 25].

At this point systems had a set of candidate suggestion grouped by, at least, location. Depending on the service used to gather candidate suggestion systems had some level of data on each candidate. Some services provided formated data such as name, ratings, popularity, comments, opening hours, etc. Some services simply provided URLs. Some systems supplemented the data they had about candidates with more information gathered from the candidate suggestion's URL and pages linked from this URL. Useful information needed was extracted from these pages [16, 12, 13]. Systems often also supplemented the example suggestion from information from their URLs or information about them provided by other services.

Many systems also filtered by time in some fashion. Some systems extracted business hours from the candidate websites or services [16, 13, 24]. Another strategy used was to use social check-ins from sites like FourSquare to estimate when venues were typically open [14]. A third strategy was to map categories to coarse business hours, for example "morning" and candidates with the same category were assigned the same business hours [8]. Candidate suggestions that had business hours that did not match with the context were then filtered out.

After filtering candidate suggestions and gathering information about them the candidates had to be ranked. Systems used various different features to rank candidates. A common feature was to use the textual similarity between example suggestions that the user found to "look interesting" and candidate suggestions. Other features used included the number of reviews, how positive the reviews were, and the rank of the candidate given by the original service it was listed in [25]. Some systems also gave a higher ranking to

candidates with the same categories as example suggestions that the used found to "look interesting" [12, 25, 24]. Another feature used by some systems was to assign each category a season and give a higher rank to candidate suggestions if its season matched the context's season [16, 8].

All the features used by systems where then combined in some fashion, often linearly, to give each candidate suggestion an overall ranking. In addition to boosting scores of candidates that matched highly rated example suggestions some systems also penalized candidates that matched poorly rated example suggestions [14, 19, 8].

After ranking the example suggestions the final step systems did was to develop a description for each suggestion in the ranked list. Some systems gathered this information directly from the text in the webpage [16], the meta description tags [12], or snippets provided by the services used to gather candidate suggestions [12]. Another option used by some systems was to pass the URL to a search engine and use the snippet that it returned [14, 8]. Finally, some systems returned positive reviews for the suggestion [25, 19].

5.5.2 2013 Participant Approaches

The task systems were required to do didn't change substantially from 2012 to 2013 and the basic strategy used by 2013 systems was similar to the strategy used by 2012 systems. Systems gathered candidate suggestions from various listings, search engines, and APIs. Different systems used different keywords to search by, but again searching by location was common in order to group candidates by context. These services provided information about the candidates which sometimes included the URL, reviews, popularity, category and other information.

One change from the 2012 experiment was that the temporal component was dropped from contexts so systems no longer spent effort to filter by opening hours or season.

Again different systems used different features to rank the list of candidate suggestions. Textual similarity between suggestions the user liked and candidates was a common features, but other features such as reviews of the suggestion, popularity, etc. were also used. How these features were combined varied by system. Some systems also introduced a attempt at diversity in the ranked results based on the category of suggestions, i.e., systems had a variety of suggestions from different categories in the results.

Like in 2012, systems also had to generate descriptions for candidates to show users. Again strategies such as used snippets from a search engine and returning reviews or descriptions from the service were used. Some systems used more complicated strategies such as developing descriptions from multiple sources.

Organization	Runs
CSIRO 1	baselineA, baselineB
CSIRO 2	csiroht, csiroth
Georgetown University	gufinal, guinit
HP Labs China	hplcranking, hplcrating
Indian Statistical Institute	watcsl2a, watcsl2b
Inst de Recherche en Info de Toulouse	iritsplit3CPv1, iritSplit3CPv2
Inst of Comp Tech, Chinese Academy of Sciences	ICTCONTEXTRUN1, ICTCONTEXTRUN2
Pattem Recognition and Intelligence System Lab	PRISabc
TNO and Radboud University Nijmegen	run01TI, run02K
University of Amsterdam	UAmsCS12wtSUM, UAmsCSl2wtSUMb
University of Delaware	udelnp, udelp
University of Delaware, Infolab	UDInfoCSTc, UDInfoCSTdc
University of Indonesia	FASILKOMUIO1, FASILKOMUIO2
University of Waterloo	waterlool2a, waterlool2b

Table 5.3: Teams that participated in the 2012 track.

Another change from the 2012 experiment was the option to use the ClueWeb12 dataset rather than the open web. Most systems did not take this option but for those that did an extra step was needed to gather candidate suggestions. Some systems simply mapped open web documents to ClueWeb12 but many systems indexed ClueWeb12 (or some subset of ClueWeb12) and retrieved candidate suggestions by issuing searches against their index.

5.5.3 Leading Approaches

In addition to looking at the general strategy used by participants, it is also interesting to look at the strategies used by groups who performed well. In 2012, the iritSplit3CPv1 run performed the best (according to P@5), their overall strategy was similar to other strategies however we can look at a couple choices made when implementing this system. Hubert and Guillaume [8] developed a model for each user that consisted of terms they liked and terms they disliked. Each suggestion was boosted if it matched the positive terms and given a lower score if it matched the negative terms.

The runs guinit and gufinal also performed quite well. Yates et al. [25] implemented a system that had heavier filtering of unwanted pages (like error pages) then other systems. They also incorporated the idea of mixing well-known attractions (ones the location is famous for) with personalized attractions. The difference between these two runs is whether the description judgment was used to match example and candidate suggestions (guinit) or whether the document judgment was used (gufinal).

Organization	Runs
Pattern Recognition and Intelligent System laboratory	1, 2
University of Waterloo	baselineA, baselineB
Georgetown University (Yang)	BOW_V17, BOW_V18
National University of Ireland, Galway	CIRG_IRDISCOA, CIRG_IRDISCOB
University of Lugano	complexScore, simpleScore
University of Indonesia	csui01, csui02
Democritus University of Thrace	DuTH_A, DuTH_B
Centrum Wiskunde & Informatica	IBCosTop1
Institt de Recherche en Informatique de Toulouse	IRIT.ClueWeb, IRIT.OpenWeb
Indian Statistical Institute	isirun
School of Information Sciences, University of Pittsburgh	ming_1, ming_2
University of Sao Paulo	run01
Institute of Computing Technology, Chinese Academy of Sciences	RUN1, RUN2
University of Amsterdam	UAmsTF30WU
University of Delaware	udel_run_D, udel_run_SD
InfoLab at University of Delaware	UDInfoCS1, UDInfoCS2
School of Information and Library Science, UNC-CH	uncsils_base, uncsils_param
University of Glasgow (Terrier Team)	uogTrCFP, uogTrCFX
Department of ITEC, YORK University, Toronto	york13cr1, york13cr2

Table 5.4: Teams that participated in the 2013 track.

Chapter 6

Conclusions & Future Work

6.1 Conclusion

During the past two years we have ran two TREC based experiments that involved the participation of several research groups. We developed the framework used to run these experiments, how to develop profiles and contexts, what the systems should produce, and how to judge the system output. Profiles were developed by asking real users for ratings of attractions, contexts were developed by sampling US cities and randomly picking the temporal component, and judging was done with a mix of professional assessors and real user judgments.

We also developed a metric that is an extension to TBG which caters to the task the systems in the TREC experiments developed. The metric takes into account the effects of positive and negative descriptions and wasting user time by having them read negative documents.

As can be seen from figures 5.1 and 5.2 our extension to TBG correlates quite well with other metrics, for example when compared to P@5 the Kendall's tau coefficient is between 0.79 and 0.90 for our experiments. The benefit from using this metric is not immediately clear and in fact, for our experiments in TREC, we treated P@5 as our primary measure used to determine which systems performed best. However the modified TBG metric has a more realistic user model and we plan to continue investigating it for both possible improvements and in experiments that support its usefulness.

6.2 Future Work

These experiments will continue for another year as part of TREC 2014. We plan to run a similar experiment to 2013 with a few changes:

- We will only use users from crowdsourcing sites.
- We will have more example suggestions in profiles and investigate if this improves system performance.
- We will investigate if systems can provide more relevant suggestions if demographic data such as age is in the profile.
- We will investigate how adding who the user is travelling with (friends, family, alone) to the context changes systems.
- We will investigate the effects of sampling from US cites with known tourism activity rather than sampling from cities based on population.

We also want to investigate our usage of crowdsourcing, our metric, and the focus on the task of only extracting all candidate suggestions from the open web and ClueWeb12.

6.2.1 Crowdsourcing

We used a mix of crowdsourced workers and university students as users in the 2013 experiment. We want to investigate how these two compare in terms of quality of judgments and difficulty in discovering interesting suggestions for. We hope to use the data already gathered to get some insight into crowdsourcing.

6.2.2 Practicality of TBG

We plan to continue using our modified TBG metric in the 2014 experiments, however we want to investigate if we could improve this metric. As part of our experiments we gathered timing data for how long users took to make judgments for the descriptions and websites.

In our experiments we used the mean time across all users as an estimate of how long every user takes to make judgments. Smucker and Clarke have discussed [21] an option for

modelling user behaviour in a more realistic way. We will investigate using their techniques and develop models for how long each user takes to make judgments, using these models in our metric calculations. This would involve picking at random how long a user takes to make a judgment out of a model of all the times the user took to make judgments. Since this involves random sampling the metric would become more complicated and we would have to report both the mean and variance for the metric. Other improvements to our metric can also be considered during future investigations.

6.2.3 Sub-collection Building

A first step most systems took when building ranked list of suggestions was to gather a list of candidate suggestions from either the open web or ClueWeb12 using a variety of services and APIs such as Yelp. In order to make it so that the differences in the systems are limited to only the differences the systems used to rank suggestions and generate descriptions we will investigate the possibility of generating a list of candidate suggestions for each context in advance that systems can use.

This end goal here will be to have a sub collection of known candidate suggestions with at least a URL for each candidate.

References

[1] Nicholas J. Belkin, Charles L.A. Clarke, Ning Gao, Jaap Kamps, and Jussi Karlgren. Report on the SIGIR workshop on "entertain me": Supporting complex search tasks. *SIGIR Forum*, 45(2):51–59, December 2012.

[2] Stefan Büttcher, Charles Clarke, and Gordon V. Cormack. *Information Retrieval: Implementing and Evaluating Search Engines*, chapter 12. The MIT Press, 2010.

[3] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 621–630. ACM, 2009.

[4] Gordon V Cormack and Thomas R Lynam. Trec 2005 spam track overview. In *Proceedings of TREC*, volume 14, 2005.

[5] Adriel Dean-Hall, Charles LA Clarke, Jaap Kamps, and Paul Thomas. Evaluating contextual suggestion. In *Proceedings of EVIA*, 2013.

[6] Donna Harman. Trec 1992 overview of the first text retrieval conference. In *Proceedings of TREC*, volume 1, 1993.

[7] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[8] Gilles Hubert and Guillaume Cabanac. Irit at trec 2012 contextual suggestion track. In *Proceedings of TREC*, volume 12, 2012.

[9] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[10] Marijn Koolen, Jaap Kamps, and Hugo Huurdeman. Contextual suggestion from wikitravel: Exploiting community-based suggestions. In *Proceedings of TREC*, volume 12, 2012.

[11] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[12] Bingyang Liu, Tong Wu, Xianghui Lin, Yanqin Zhong, Qian Liu, Yue Liu, and Xueqi Cheng. Ictnet at context suggestion track trec 2012. In *Proceedings of TREC*, volume 12, 2012.

[13] Abhishek Mallik, Mandar Mitra, and Kripabandhu Ghost. Contextual suggestion. In *Proceedings of TREC*, volume 12, 2012.

[14] David Milne, Paul Thomas, and Cecile Paris. Finding, weighting and describing venues: Csiro at the 2012 trec contextual suggestion track. In *Proceedings of TREC*, volume 12, 2012.

[15] Alistair Moffat and Justin Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems (TOIS)*, 27(1):2, 2008.

[16] Lin Qiu, JunRui Peng, QianQian Wang, Yue Liu, ZhiHua Zhou Weiran Xu, Guang Chen, and Jun Guo. Pris at trec2012 contextual suggestion track. In *Proceedings of TREC*, volume 12, 2012.

[17] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[18] Stephen E Robertson and Ian Soboroff. The trec 2002 filtering track report. In *Proceedings of TREC*, volume 11, 2002.

[19] Maya Sappelli, Suzan Verberne, and Wessel Kraaij. Tno and run at the trec 2012 contextual suggestion track: Recommending personalized touristic sights using google places. In *Proceedings of TREC*, volume 12, 2012.

[20] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[21] Mark D Smucker and Charles LA Clarke. Modeling user variance in time-biased gain. In *Proceedings of the Symposium on Human-Computer Interaction and Information Retrieval*, page 3. ACM, 2012.

[22] Mark D Smucker and Charles LA Clarke. Time-based calibration of effectiveness measures. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 95–104. ACM, 2012.

[23] Ellen M Voorhees et al. The trec-8 question answering track report. In *Proceedings of TREC*, volume 8, 1999.

[24] Peilin Yang and Hui Fang. An exploration of ranking-based strategy for contextual suggestion. In *Proceedings of TREC*, volume 12, 2012.

[25] Andrew Yates, Dave DeBoer, Hui Yang, Nazli Goharian, Steve Kunath, and Ophir Frieder. (not too) personalized learning to rank for contextual suggestion. In *Proceedings of TREC*, volume 12, 2012.

[26] Emine Yilmaz, Milad Shokouhi, Nick Craswell, and Stephen Robertson. Expected browsing utility for web search evaluation. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1561–1564. ACM, 2010.