CWI and TU Delft at TREC 2013: Contextual Suggestion, Federated Web Search, KBA, and Web Tracks

Alejandro Bellogín[†], Gebrekirstos G. Gebremeskel[†], Jiyin He[†], Jimmy Lin[*], Alan Said[†], Thaer Samar[†], Arjen P. de Vries[†‡], Jeroen B. P. Vuurens[‡]

[†]Centrum Wiskunde en Informatica, Amsterdam, The Netherlands
[‡]Delft University of Technology, Delft, The Netherlands
[*]University of Maryland, College Park, Maryland, USA

{alejandr,gebre,he,alan,samar,arjen}@cwi.nl
j.b.p.vuurens@tudelft.nl, jimmylin@umd.edu

ABSTRACT

This paper provides an overview of the work done at the Centrum Wiskunde & Informatica (CWI) and Delft University of Technology (TU Delft) for different tracks of TREC 2013. We participated in the Contextual Suggestion Track, the Federated Web Search Track, the Knowledge Base Acceleration (KBA) Track, and the Web Ad-hoc Track. In the Contextual Suggestion track, we focused on filtering the entire ClueWeb12 collection to generate recommendations according to the provided user profiles and contexts. For the Federated Web Search track, we exploited both categories from ODP and document relevance to merge result lists. In the KBA track, we focused on the Cumulative Citation Recommendation task where we exploited different features to two classification algorithms. For the Web track, we extended an ad-hoc baseline with a proximity model that promotes documents in which the query terms are positioned closer together.

1. INTRODUCTION

We have participated in four tracks this year. The Contextual Suggestion track is introduced in Section 2, where our personalised approach on top of the ClueWeb12 collection is presented. Section 3 shows our results on the Federated Web Search track, where we participated in the two available tasks: resource ranking and results merging. Our participation in the Knowledge Base Acceleration is described in Section 4, extending our participation of last year with further experiments and new approaches. Finally, a proximity model applied to the Ad-hoc Web track is presented in Section 5.

2. CONTEXTUAL SUGGESTION

In this section we describe our work in the contextual suggestion track, which represents our first participation in this track.

2.1 Track Description

The Contextual Suggestion Track investigates search techniques for complex information needs that depend on context and user interests. Input to the task are a set of profiles (*users*), a set of example suggestions (*attractions*), and a set of contexts (*locations*). Each attraction includes a title, a description, and an associated URL. Each profile corresponds to a single user, and indicates the user's preference with respect to each attraction using two ratings: one for the attraction's title and description and another for the attraction's website. Finally, each context corresponds to a particular geographical location (a city and its corresponding state in the United States).

For each pair of context and profile, a ranked list of up to 50 ranked suggestions (attractions) should be generated. Each suggestion should be appropriate to both the user profile and the context. The description and title of the suggestion may be tailored to reflect the preferences of that user.

The source from where the suggestions are selected may be either the ClueWeb12 collection or the open web. Our submission was based on the former, in order to ensure reproducibility and further comparison of our results.

2.2 Initial Ideas

At the beginning we aimed to use approaches based on Collaborative Filtering [27]. However, after taking a closer look into the problem, we realised that the items (documents) for which we have some information – i.e., the rated attractions – correspond to a context different to any of the target contexts. That is, whereas the ratings were given in the context of Philadelphia, PA, the contexts for which suggestions have to be generated are other metropolitan areas different to that one. Therefore, they cannot be used directly – since the potential candidates naturally depend on the context – which limits the scope of these techniques since they are based on the *word of mouth* effect. For this reason, we decided to explore content-based techniques [21], which represent users and items in the same space, and generate recommendations based on some distance measure. We discuss our approach in Section 2.3.

We considered additional methods to exploit the ratings, but we were not able to generate a valid submission based on them. For instance, we also applied neighbour-based methods [11] to find suggestions based on the documents similar (not the documents liked or rated, as in standard collaborative filtering) to the user's neighbours; these neighbours may be found by exploiting textual or rating-based similarity between user profiles. Besides, a separate learning for the positive and negative aspects of each user profile was also considered, and will be explained in Section 2.3.6.

2.3 Methodology

Our approach is to compare text of user's attractions with documents that mention a context. The first step we did in this task was to find all documents from the ClueWeb12 collection that mention each context. In parallel, we generated user profiles based on the descriptions of the attractions rated by them. Finally, we used the cosine similarity between the context documents and the user

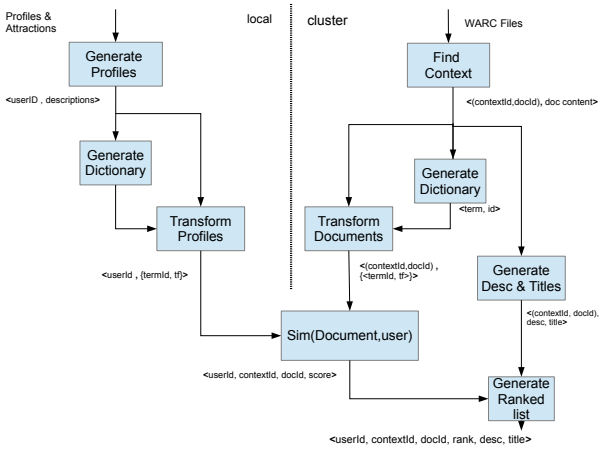

Figure 1: Contextual Suggestion Track. Workflow followed to generate contextual suggestions.

Input: ClueWeb12 WARC files, contexts: <contextId, (city, state)>
Output: <(contextId, docId), doc Raw Content> pairs
begin
  extract document id: docId
  content ← HTML content of the map input value
  for *context* ∈ *contexts* **do**
    if *context* ∈ *content* **then**
      if *(city, ??)* ∉ *content* **then**
        `??  stands for other state different than the target one`
        outKey ← (contextId, docID)
        outVal ← content
        emit(outKey , outVal)
      end
    end
  end
end

Algorithm 1: Contextual Suggestion Track. Pseudo-code for *finding contexts*

profiles, as represented in the $|V|$-dimensional vector space, where each element in the vector is a pair of term id and the frequency of that term in the document, $|V|$ is the size of the vocabulary [28]. Based on these similarity values, we ranked the documents tailored to each user. Figure 1 shows the complete procedure that we followed to generate a ranked list of documents for the (context, user) pairs. In the following sections we discuss our approach in detail.

2.3.1 Document Filtering (Finding Contexts)

The first step in the pipeline presented in Figure 1 consists of finding the most appropriate documents related to a context. For this task, we are interested in maximising the precision of this filtering, instead of the recall – that is, we want to be confident enough that the filtered documents mention the specific target context, although in this process we may lose some (potentially relevant) documents due to typographical or parsing errors. With this goal in mind, we focused on extracting the relevant documents for each context from the ClueWeb12 collection and created smaller sub-collections. Thanks to this filtering, the next steps – i.e., the ranking generation – can be executed using a smaller set of documents (see Table 1) which in fact allows for more efficient processing.

Sub-collections were made as follows. A MapReduce job read the document content from the entire collection and kept only those documents that mention exactly the context as provided by the organisers, ignoring those documents where the target city appeared more than once, but with different states. We decided not to keep such documents as they could (potentially) consist of lists of city names, which we believe would provide zero interest to any user. To do this, we used a regular expression to check the mention of contexts in the document – that is, the pair (city, state) mentioned above –, along with another regular expression checking if the city was mentioned near another state different from the target state. For example, for the context *Springfield, IL*, we would include in its corresponding sub-collection all the documents where *Springfield* and *IL* are mentioned and only spaces or commas are in between, however, a document would not be valid if, besides *Springfield, IL*, it also contains *Springfield, FL*. Algorithm 1 shows the pseudo-code of this step.

After filtering, we found $13,548,982$ documents that mention at least one of the contexts among the total number of $733,019,372$ documents from the ClueWeb12 collection. Table 1 shows the number of documents found for each context, along with the intersection of these documents with those provided by the organisers as track sub-collection[1].

2.3.2 Modelling Documents

In this section, we describe how we transformed the candidate documents in each sub-collection into its representation in the Vector Space Model (VSM). First, we generated a dictionary that has a mapping between terms and their integer ids. To decrease the size of the dictionary and remove useless terms, we filtered out the HTML tags from the content of the documents, then we removed stop-words and non-alphanumeric terms. Algorithm 2 shows the pseudo-code for generating the collection's dictionary. After that, we used this dictionary to transform documents into vectors of weighted terms, where the weight of each dimension (term) is the standard term frequency *tf*. We implemented this process as a Map-Reduce job that reads the ClueWeb12 WARC files and transforms them into vectors of pairs, where each pair is the term id and its corresponding frequency. Algorithm 3 gives specific details on this job. Since now we deal with a representation based on integer instead of string vectors for each sub-collection of documents, the size of the sub-collections will decrease and a faster processing will be possible. Table 2 shows the effect on the collection size after cleaning, transforming, and optimising the vector-based document representation.

2.3.3 Modelling Users

We generate each user's profile according to the user's preference for the given attractions and the descriptions of those attractions. The initial idea was to use the content of the attractions instead, by extracting the HTML content of the attractions websites. However, we found a coverage problem between the ClueWeb12 collection and this set of attractions: first, only 7 pages were found with a one-one URL mapping, which ended up to 35 by matching hostname and considering URL variations such as adding or removing *www* and *http(s)*; second, we found that the user ratings for attraction's descriptions and websites were very similar in most of the cases. Third, to participate as a ClueWeb12 submission we could not crawl the attractions from the Open Web or use any other external information. Figure 2 shows a histogram of the difference

[1] https://sites.google.com/site/treccontext/trec-2013/subcollection

Table 1: Contextual Suggestion Track. Number of documents per context in our sub-collections, in the provided one by the organisers (given), and the intersection (and ratio) between them.

Context	Number of docs	given	intersect	Ratio
Springfield, IL	138996	775	206	26.58
Cheyenne, WY	105864	599	190	31.72
Fargo, ND	160525	665	193	29.02
Kennewick, WA	51675	414	106	25.60
La Crosse, WI	74479	539	122	22.63
Valdosta, GA	61196	423	125	29.55
Houma, LA	40146	246	62	25.20
Greenville, NC	81836	956	82	8.58
Hickory, NC	67752	796	107	13.44
Cincinnati, OH	586429	880	306	34.77
St. Louis, MO	633701	992	339	34.17
Asheville, NC	313093	817	395	48.35
Beckley, WV	45116	390	114	29.23
Myrtle Beach, SC	201153	696	329	47.27
Orlando, FL	912922	1125	377	33.51
Washington, D. C., DC	4165769	1456	571	39.22
Anniston, AL	42111	377	90	23.87
Crestview, FL	13728	499	78	15.63
Youngstown, OH	84688	436	107	24.54
Macon, GA	140537	565	174	30.80
Monroe, LA	72972	633	88	13.90
Tampa, FL	637727	1097	355	32.36
Albany, NY	349842	1122	307	27.36
Sumter, SC	32670	548	72	13.14
Wenatchee, WA	57243	419	106	25.30
Lakeland, FL	106887	472	161	34.11
Appleton, WI	102026	525	185	35.24
Lewiston, ID	30373	499	96	19.24
Lima, OH	50061	585	107	18.29
Rochester, NY	429068	1044	323	30.94
Gulfport, MS	55055	400	150	37.50
Johnson City, TN	64378	826	129	15.62
Lynchburg, VA	101923	472	126	26.69
Atlanta, GA	1587486	1161	550	47.37
Williamsport, PA	74581	467	128	27.41
Corpus Christi, TX	172425	821	191	23.26
Dothan, AL	67954	372	106	28.49
Parkersburg, WV	46826	531	135	25.42
Wichita, KS	264770	687	242	35.23
Greenville, SC	230483	793	200	25.22
Yakima, WA	85636	530	159	30.00
Cedar Rapids, IA	138705	581	193	33.22
Kahului, HI	26491	667	83	12.44
Harrisburg, PA	243803	743	298	40.11
Bismarck, ND	106302	494	158	31.98
Saint George, UT	15686	671	45	6.71
Montgomery, AL	185157	693	185	26.70
Palm Bay, FL	33922	1064	69	6.48
Rockford, IL	160996	435	197	45.29
Manhattan, KS	95818	698	152	21.78
	13548982	33696	9369	

between the ratings for descriptions and websites – i.e., a negative value denotes that the rating for the website is higher than for the description. We can observe that most of the values are concentrated around 0, which means that no difference is observed between the two ratings. We have to note, however, that this pattern may change depending on the actual attraction analysed, as in Figure 3 where a larger shift is observed for the attractions with id 52 (*Eastern State Penitentiary*) and 57 (*Chinatown*).

Table 2: Contextual Suggestion Track. Effect of optimisation of the vector representation on the size of the collection.

Dataset	Size (GB)
Subcollection	918
Vector representation	40

Data: ClueWeb12 WARC files, stop_words (from distributed cache)
Input: <(contextId,docId), WARC record>
Output: <term, termId>
begin mapper
  content ← HTML content of the map input value;
  cleaned_content← jsoup.clean(content);
  tokens← tokenize(cleaned_content);
  for *each tok* ∈ *tokens* **do**
    docTerms ← Map<String,Long> ;
    if *tok* ∉ *stop_words* **then**
      if *tok* ∉ *docTerms* **then**
        docTerms.add(term,1);
      end
    end
  end
  for *each term* ∈ *docTerms* **do**
    emit(term,1);
  end
end
begin reducer
  termIdMap ← Map<String,Integer>;
  if *term* ∉ *termIdMap* **then**
    termIdMap.add(term,id);
  end
  for *each term* ∈ *termIdMap* **do**
    emit (term, id);
  end
end

Algorithm 2: Contextual Suggestion Track. Pseudo-code for *generating collection dictionary*

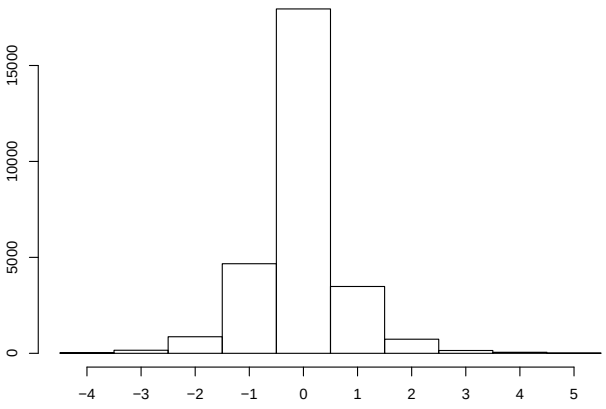

Figure 2: Contextual Suggestion Track. Histogram for rating differences considering all the contexts (description rating - website rating).

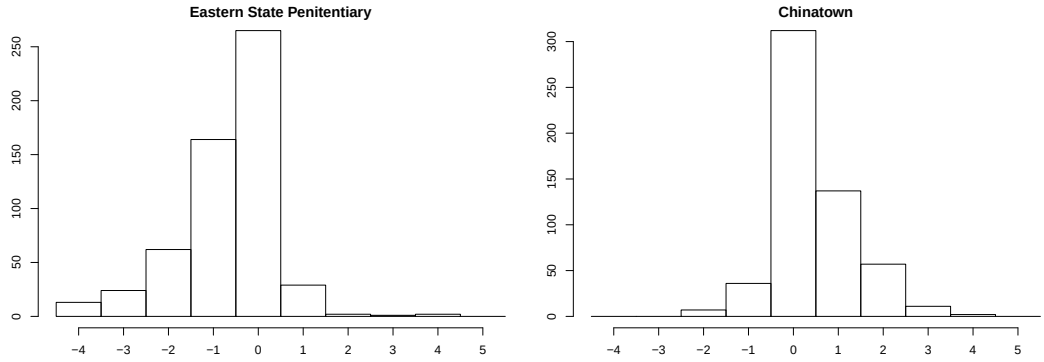

Figure 3: Contextual Suggestion Track. Histogram for rating differences for two attractions.

```
Data: ClueWeb12 WARC files, termIdMap<term,id>
      (dictionary from distributed cache)
Input: <(contextId,docId), WARC record>
Output: <(contextId, docId), vector(terms,tf)>
begin
    content ← HTML content of the map input value;
    cleaned_content ← jsoup.clean(content);
    tokens ← tokenize(cleaned_content);
    Map<Integer,Integer> : docTermsFreqMap<termId,tf>;
    for each token ∈ tokens do
        termId ← termIdMap.get(token);
        if termId ∉ docTermsFreqMap then
            docTermsFreqMap.put(termId,1) ;
        else
            docTermsFreqMap.get(termId)++ ;
        end
    end
end
```

Algorithm 3: Contextual Suggestion Track. Pseudo-code for *transforming documents into vectors*

Inspired by the best approaches of last year [18, 35], we used three different methods to represent the users profiles:

- generate user profiles based only on the attraction descriptions, without taking into account the ratings
- generate positive user profiles based on the attraction descriptions that have positive ratings from each user
- generate negative user profiles based on the attraction descriptions that have negative ratings from each user

Since the ratings are on 5-point scale, each rating represents a user's level of interest in visiting the corresponding attraction, the levels ranging from "0" for strongly uninterested to "4" for strongly interested. In this context, we consider the "2.5" as threshold between negative and positive ratings.

Moreover, we need the same representation between the user profiles and the candidate documents. Because of this, we transformed these profiles into weighted vectors following the same methodology we used to transform the documents (see Section 2.3.2).

2.3.4 Personalising Ranking

To generate the final ranking (given a pair of context and user information), we computed the similarity in the vector space representation between the document and user profile representations, as presented before in Sections 2.3.2 and 2.3.3. With this goal in mind, we tested the Jaccard and cosine functions as similarities, which are defined like:

$$sim(u,d) = \frac{|u \cap d|}{|u \cup d|} = \frac{\sum_i \delta(u_i == d_i! = 0)}{\sum_i \delta(u_i! = 0 || d_i! = 0)} \quad (1)$$

$$sim(u,d) = \cos(u,d) = \frac{\sum_i u_i \cdot d_i}{\sqrt{\|u\|_2}\sqrt{\|d\|_2}} \quad (2)$$

where δ is the function that outputs 1 if its argument is true. Note that since Jaccard does not take into account the actual value encoded in the vector (in our case, frequencies), we binarise the vectors prior computing the Jaccard similarity.

We implemented a MapReduce job to compute these similarities, in part because they involve a large dimensional vocabulary space, mainly produced by the document representation. This MapReduce job only involves map tasks, where the user vectors are added to the distributed cache, and then, when the similarity is computed, it outputs the tuple (context, document, user) as key and the actual similarity as value.

2.3.5 Generating Description and Titles

As mentioned before, besides producing a document ranking for each user and context, we also have to generate a description and title for each suggestion, and if possible, tailored to the users, that is, considering their preferences as means to explain why such document is presented.

We decided to only provide personalised descriptions, since we consider the title as a global property of the document, inherent to its content and, thus, should not be different for each user. In this situation, we generated the titles by extracting the title or heading tags from the HTML content of the document. On the other hand, we observed the task of generating descriptions similar to snippet generation where the query is the combination of context and user preferences [22]. Because of that, we aimed at obtaining the most relevant sentences for the user within the document in a particular context. To do this, we first split the document into sentences by using the Java BreakIterator class[2] which can detect sentence boundaries in a text. We then followed similar steps to those of the document ranking but at a sentence level, i.e., filter out those sentences not mentioning the context and ranking the remaining sentences according to their similarity with the user profile. Finally, we assumed that larger descriptions were preferred, and hence, we concatenated sentences – in decreasing order of similarity – until

[2] http://docs.oracle.com/javase/6/docs/api/java/text/BreakIterator.html

the maximum number of bytes 512 was reached, controlling to not combine two very similar sentences to decrease the redundancy.

2.3.6 Sub-Collection Post-Filtering

In this section, we present a post-filtering method we ran on each sub-collection. Due to temporal and technical issues, we were not able to submit the results after applying this filtering; we analyse in more detail its potential effect in Section 2.4.

The main motivation to perform this post-filtering was the fact that a webpage mentioning a context does not provide enough evidence that such webpage is about an attraction worth to visit, and thus, whether it should be recommendable at all. To address this problem we built a classifier based on the content of selected websites from ClueWeb12. These selected websites aimed to capture well-known travel and tourism related pages, in particular we considered: yelp, tripadvisor, wikitravel, zagat, xpedia, orbitz, and yahoo-travel. We used a decision tree classifier as implemented by Weka[3], where the positive labels correspond to every page under the domains aforementioned (we found 171,188 documents), whereas the negative labels were assigned based on a random number of pages not contained in the previous positive sub-collection. This is a typical unbalanced classification problem (or imbalanced dataset) [34], where one class (the negative one, in our case, not attraction) has a much larger number of examples than the other one, which is, in fact, the class of interest. To allow a fair learning by the classifier, we decided to not account for this bias and selected the same number of negative documents as positive, something known as *subsampling* in Machine Learning; we have to note, however, that other strategies exist in the literature [3].

Once the classifier was built, we labelled each document in our context-dependent sub-collections as either positive (*similar* to webpages of the selected travel and tourism related sites) or negative, and perform the same ranking approach presented in Section 2.3.4 but only with the positive documents.

2.4 Analysis

Now we analyse the approach we submitted and how it compares to other approaches that we wanted to try but were unable to submit on time (see Section 2.2). We first discuss the effect of the sub-collection creation may have in our results, and then we experiment with other similarity and formulations for the ranking step.

2.4.1 Effect of the Sub-Collection

As mentioned in Section 2.3.1, the organisers provided a sub-collection from ClueWeb12 specifically tailored for this task[4]. The first analysis we present is a comparison between the submitted ranking (based on filtering the entire ClueWeb12 collection) and the same algorithm (Section 2.3.4) using the given sub-collection. In this way, we can discriminate which part has a larger effect in the final result, the filtering method to create sub-collections or the actual ranking function.

The results are presented in Table 3. We present the MRR metric as computed by the evaluation script provided by the organisers. However, since we are also evaluating unsubmitted runs, we should not use the judgements for the descriptions, since eventually the descriptions could be different for every method, and thus their relevance may differ (in contrast with the document relevance, which we assume to be more consistent and stable). Hence, MRR_d and P@5_d represent the performance of these methods when the description judgements are ignored (d here stands for document). Additionally, $\text{P@5}_{d\bar{g}}$ shows the value obtained when the geographical assessments are not considered.

[3] http://www.cs.waikato.ac.nz/ml/weka/

[4] As described by the organisers: "this sub-collection was created by issuing a variety of queries targeted to the Contextual Suggestion track on a commercial search engine."

Table 3: Contextual Suggestion Track. Performance of our method when the entire ClueWeb12 collection or the given sub-collection are used.

Method	MRR	MRR_d	P@5_d	$\text{P@5}_{d\bar{g}}$
IBCosTop1	**0.0559**	0.0745	**0.0587**	**0.2202**
IBCosTop1 (given)	0.0528	**0.0955**	0.0484	0.0780
With classifier	0.0090	0.0179	0.0045	0.0260
With classifier (given)	0.0256	0.0540	0.0136	0.0159

We observe in this table that the documents originally retrieved were relevant for the user but not geographically appropriate, since the value of $\text{P@5}_{d\bar{g}}$ improves significantly. This is in contrast with what happens when using the given sub-collection for this method, mostly because the documents were tailored to be appropriate for each context. Additionally, the performance in terms of MRR is higher when using the given sub-collection than our filtered sub-collections from the entire ClueWeb12 collection. This is a clear evidence that the filtering step should be refined.

As already presented in Table 1, and now in more detail in Table 4, the intersection between our sub-collections and the given one is very low. This is very clear when we use the classifier presented before as a post-processing step, since we were not able to submit this run, and thus, most of the documents recommended by this approach (since its intersection with our other method is very low) remain unjudged. This, in turn, decreases the final performance of the method based on the classifier and the complete collection. In fact, the performance improves again when the given sub-collection is used, although now the values are lower than before. This may also be due to a low coverage of the relevance assessments and further analysis is needed in this regard.

Table 4: Contextual Suggestion Track. Intersection measures between submitted and unsubmitted approaches. $x \setminus y$ represents the set difference or relative complement, i.e., those elements in x that are not y. Jacc stands for Jaccard similarity.

Method a	Method b	$\lvert a \setminus b\rvert/\lvert a\rvert$	$\lvert b \setminus a\rvert/\lvert b\rvert$	$Jacc(a,b)$
IBCosTop1	IBCosTop1 (given)	0.99	0.98	0.01
IBCosTop1	Classifier	0.95	0.86	0.04
Classifier	Classifier (given)	0.99	0.80	0.01

2.4.2 Effect of the Similarity Function

As we mentioned at the beginning in Section 2.2, one of our first ideas was to exploit collaborative filtering techniques for this task. However, as already discussed, this is not possible with the data available. Nonetheless, inspired by the nearest-neighbour techniques from the recommender systems literature [11], we developed some approaches able to capture the concept of collaborative recommendation. Basically, we start with a generated set of rankings $\{R_{u_i}\}_i$ respectively assigned to user u_i. Then, for every user u we find a set of users (neighbours) V_u based on which we will generate the final recommendations. To aggregate the rankings R_v

for $v \in V_u$ we used Borda aggregation, a standard technique from rank aggregation literature [12]. In this setting, there are two parameters we have tested: the amount of neighbours (size of $|V_u|$) and how the neighbours are selected (user similarity metric).

In Table 5 we show the best results we found using these methods. We observe that we never outperform the results of the submitted run, this may be due, like before, to a low coverage of the relevance assessments, since these approaches were not evaluated. Regarding the best parameters, 5 neighbours obtains the best results (we also tested with 10 and 50 neighbours), and the best similarity metric depends on the actual method used: when IBCosTop1 is used, the Pearson's correlation similarity based on the ratings given by the users to the example attractions achieves the best results; on the hand, when the input uses the post-filtering based on the classifier, best results are obtained with the Jaccard similarity between the textual representation of the users.

Table 5: Contextual Suggestion Track. Performance of variations of our method. NN stands for Nearest Neighbour.

Method	MRR	MRR_d	$P@5_d$	$P@5_{d\bar{g}}$
IBCosTop1	**0.0559**	**0.0745**	**0.0587**	**0.2202**
IBCosTop1 + 5NN text cos	0.0455	0.0562	0.0330	0.1486
IBCosTop1 + 5NN text Jacc	0.0433	0.0521	0.0330	0.1294
IBCosTop1 + 5NN rating cos	0.0429	0.0553	0.0349	0.1477
IBCosTop1 + 5NN rating Pearson	0.0450	0.0580	0.0358	0.1560
Classifier + 5NN text cos	0.0045	0.0112	0.0036	0.0251
Classifier + 5NN text Jacc	0.0045	0.0121	0.0045	0.0260
Classifier + 5NN rating cos	0.0045	0.0090	0.0027	0.0242
Classifier + 5NN rating Pearson	0.0045	0.0067	0.0018	0.0233
Positive profile	0.0396	0.0588	0.0359	0.1498
Negative profile	0.0045	0.0045	0.0009	0.0152
Positive + 5NN text cos	0.0426	0.0572	0.0341	0.1399

Additionally, we also tested profiles based only on the attractions with positive ratings [18, 35]. As a sanity check, we found that the recommendations based on the negative profile obtain very low performance. Interestingly, recommendations based only on the positive profile are competitive, although not as good as the submitted approach. In any case, we confirm that the collaborative nearest-neighbour approach also reduces the performance for this method in most of the situations.

2.5 Discussion

In this first attempt of the track we have faced several challenges: dealing with a very large dataset like the ClueWeb12 collection, filtering it to make it more manageable, personalising the ranking in different ways, and post-filtering the results to produce more *touristic* results. Based on our performance, there is enough room to improve in most of these steps. The filtering step, however, seems to be a determining factor in the subsequent chain of events.

While doing the aforementioned analysis, we identified a subset of documents that were submitted as part of the ClueWeb12 collection whose corresponding URLs were also submitted (by other participants) as Open Web documents. We noticed that the subjective assessments (document judgements) are not consistent, especially in the number of 3's, 4's, and -2's received by the same document in each dataset. This fact may indicate a bias towards higher performance for the methods using documents from the Open Web.

3. FEDERATED WEB SEARCH

In this section we describe our work in the federated web track.

3.1 Track Description

The Federated Web Search track investigates techniques for the selection and combination of search results from a large number of real on-line web search services. The goal of this track is to evaluate approaches to federated search at very large scale in a realistic setting, by combining the search results of existing web search engines. This year, two tasks were available: resource selection (selecting the search engines that should be queried) and results merging (combining the results into a single ranked list). We participated in both tasks and present our contribution in the next sections.

3.2 Resource Selection Task

The input for this task is a collection provided by the organisers (*FedWeb 2013 collection*) consisting of sampled search results from 157 search engines. Then, for every query the system should return a ranking such that the most appropriate search engines are ranked highest without using the actual results for the given query (which were, in fact, provided after the submission deadline of this task).

We used two different strategies to produce the resource ranking, then a third submission was generated based on their combination.

ODP based run. The Open Directory Project (ODP)[5] is a human-edited directory of the Web, constructed and maintained by volunteers. In this directory, each URL has a category assigned, which can contain sub-categories. Besides, it has a service where it returns a list of categories in response to a query. We used this service[6] to get the categories associated to each resource and to every query. We then computed similarities between the two lists of categories using cosine and Jaccard similarities (see Section 2.3.4), considering and ignoring the ranking information from each category. For the queries, we also experimented with using the actual query text in the computation of the similarity. We did not observe any significant difference between these variations, so we will only report the results for the simplest alternatives (i.e., no order, no query text).

Retrieval model based run. This strategy concatenates all the snippets from each resource and indexes them as a single document, so that when a query is issued, the aggregated documents (resources) are ranked according to their relevance with respect to the query. We built a different Lucene[7] index for each retrieval model we tested: a simple TF-IDF method, BM25 (with varying parameters k_1 and b), two language model approaches (Jelinek-Mercer with parameter λ and Dirichlet with μ). Besides, since each snippet has both a title and a description, we tested considering only the title field to match the query, only the description field (desc), or both.

Hybrid run. This run takes two rankings as input for each query. Then it aggregates the information using a Borda voting mechanism [12], where each document gives a number of votes inversely proportional to its ranking (the higher the position, the lower the number and the larger the number of votes). Finally, documents are sorted according to the number of votes (the higher, the better).

3.3 Results Merging Task

For the result merging task, the input consists of the ranked list of documents retrieved using the 157 different search engines. Then, for each query an aggregated ranked list of these documents should be returned. Documents are ranked in descending order of their relevance to the query.

[5] http://www.dmoz.org

[6] http://www.dmoz.org/search?q=QUERY&type=more

[7] http://lucene.apache.org

We employed four different strategies to aggregate the ranked lists, with different motivations.

Relevance based run (CWI13IndriQL). This run only considers the relevance of a document to a query. We rank the documents with the simple query likelihood model. For a document d and query q, the query likelihood of $p(q|d)$ is computed as

$$p(d|q) \propto \prod_{w \in q} p(w|d), \quad (3)$$

where $p(w|d)$ is computed based on the title and snippets of the document. Dirichlet smoothing with $\mu = 2500$, i.e., default setting implemented with Indri is used.

Cluster based run (CWI13clTODPJ). We take a two-step process to rank the documents. First, the resources are ranked based on one of the strategies described above (specifically, the submitted run is based on a hybrid method between ODP with Jaccard and the TF-IDF retrieval model). Then, documents within a resource are ranked by the query likelihood model.

With this run, we are interested in whether some resources are better than others in satisfying a particular information need. That is, whether the majority of documents provided by a good resource would be relevant to the query. This run was not submitted.

Diversity based run (CWI13iaTODPJ). This run is based on a different assumption. That is, by diversifying documents from different resources, it is more likely that at least one type of documents (resource) will satisfy the information need.

We take an approach similar to the cluster-based diversification [16, 17] methods, where each resource can be seen as a "cluster", as described in our cluster based run. An initial ranked list is retrieved with the query likelihood model. It is then re-ranked with an IA-select [1] based diversification algorithm.

The key elements used in the algorithm can be reduced to the following quantities: i) $V(d|q;z)$, the probability that d is relevant to q when the associated resource (cluster) is z; and ii) $p(z|q)$, the probability resource z is related to q. We compute $p(z|q)$ by normalizing the resource ranking scores discussed above (i.e., ODP + Jaccard) over all resources. To compute $V(\cdot)$, intuitively, we can set $V(d|q;z) = 1\ if\ d \in z$; otherwise 0. However, this results in a product of 0 in many cases given the IA-select framework. To address this issue we applied a minor smoothing over the $V(\cdot)$ scores which assigns a probability mass of 0.01 to cases where $d \notin z$.

Boost documents based on resource (CWI13bstTODPJ). For this run, we intend to boost the documents coming from a reliable resource. Specifically, we rank documents in descending order of

$$p(d|q,z) \propto p(d|q)p(q|z), \quad (4)$$

assuming d is independent of z given q. We then compute $p(d|q)$ as in the query likelihood run, and $p(q|z)$ as in the resource ranking, based on ODP and Jaccard similarity.

3.4 Results

In addition to the officially submitted runs, we also evaluated all the variants of the methods presented above. The 2012 data set is described in [25] and although the content is similar, we have to note that the search results correspond to the year 2012 (*FedWeb 2012*) and the search engines included are not exactly the same as in the FedWeb 2013 collection.

3.4.1 Resource Selection

For the resource selection task we tested different variations of the strategies presented above. Table 6 shows the results obtained for some of these methods with the FedWeb 2012 collection. These values were computed generating relevance judgements for each resource according to the relevance judgements (based on documents) included in the collection for a subset of 50 TREC queries. We aimed to emulate the procedure described in the track page, where the organisers note that *the relevance of a search engine for a given query is determined by calculating the graded precision on the top 10 results*, using weights according to the relevance levels of the documents: Nav and Key levels are assigned a weight of 1, Hrel a value of 0.5, and Rel, 0.25.

Table 6: Federated Web Track. Performance of some variations of our approaches for the resource selection task using the FedWeb 2012 collection (ordered by MAP).

Method	MAP	nDCG	MRR
TF-IDF+ODP Jacc	**0.338**	**0.516**	0.564
TF-IDF	0.285	0.412	**0.610**
ODP Jaccard	0.283	0.471	0.439
BM25 (1.2, 0.2)	0.283	0.400	0.545
LM ($\lambda = 0.1$)	0.280	0.407	0.590
ODP Cosine	0.278	0.462	0.400
BM25 (1.2, 0.8)	0.272	0.397	0.557
LM ($\lambda = 0.5$)	0.263	0.394	0.571
LM ($\lambda = 0.9$)	0.252	0.387	0.566
LM ($\lambda = 0.1$) desc	0.241	0.386	0.602
LM ($\mu = 200$)	0.240	0.378	0.551
LM ($\mu = 2000$)	0.240	0.378	0.551
BM25 (1.2, 0.8) desc	0.239	0.383	0.608
TF-IDF title	0.215	0.321	0.495

We found that the best methods (in FedWeb 2012) were the TF-IDF retrieval method (where the query is issued on the description and title fields), the Jaccard similarity over ODP categories, and a combination of these approaches. Moreover, methods where the query is applied only to one the fields had a much lower performance. We present in the table only the best values for each of them (Jelinek LM for the description field and TF-IDF for the title) and an additional method (BM25 desc) which will serve us as reference later.

Table 7: Federated Web Track. Performance of the variations of our approaches in the resource selection task (ordered by nDCG@20).

Method	Run	nDCG@20	ERR@20
BM25 (1.2, 0.8) desc	-	**0.1588**	**0.0204**
LM ($\lambda = 0.1$) desc	-	0.1476	**0.0204**
BM25 (1.2, 0.2)	-	0.1346	0.0068
LM ($\lambda = 0.1$)	-	0.1322	0.0068
TF-IDF	CWI13SniTI	0.1235	0.0067
BM25 (1.2, 0.8)	-	0.1223	0.0102
LM ($\lambda = 0.5$)	-	0.1218	0.0051
LM ($\lambda = 0.9$)	-	0.1153	0.0041
LM ($\mu = 2000$)	-	0.1033	0.0051
LM ($\mu = 200$)	-	0.1017	0.0051
TF-IDF title	-	0.1016	0.0017
TF-IDF+ODP Jacc	CWI13ODPTI	0.0961	0.0034
LM ($\lambda = 0.9$)	-	0.0934	0.0017
ODP Jaccard	CWI13ODPJac	0.0497	0.0000

The results obtained, however, with the FedWeb 2013 collection are completely different (see Table 7). The three runs we submitted performed not consistently with respect to what we observed in the training collection, where, for instance, the hybrid approach was the best one. Furthermore, some of the other methods evaluated had a much higher performance values, in particular the use of the description as the only field to issue the queries turned out to be the most effective approach for the FedWeb 2013 collection, and one of the worst methods in the FedWeb 2012 dataset.

3.4.2 Results Merging

As presented before, we experimented with one run based on document relevance and with three other runs depending on the output of the previous task, that is, a ranking of resources. We used the hybrid method (submitted run CWI13ODPTI) because it was the best performing method in the FedWeb 2012 collection, as we analysed before, and thus we expected it to perform equally well in the FedWeb 2013 collection, in particular now, to aggregate search results.

Table 8: Federated Web Track. Performance of our approaches ordered by P@10. * indicates an unsubmitted run.

Method	P@10	nDCG@20	nDCG@50	nDCG
2013 data				
CWI13bstBM25desc*	**0.3408**	0.1224	0.2024	0.5366
CWI13IndriQL	0.3220	**0.1622**	**0.2371**	**0.5438**
CWI13iaTODPJ	0.2840	0.1509	0.1915	0.5253
CWI13bstTODPJ	0.2500	0.1466	0.1839	0.4973
CWI13clTODPJ*	0.1940	0.0551	0.0892	0.4610
2012 data				
CWI12bstTODPJ*	**0.4960**	0.1246	0.1989	0.6081
CWI12IndriQL*	0.4900	**0.1464**	**0.2627**	**0.6525**
CWI12clTODPJ*	0.2200	0.0666	0.1106	0.5462
CWI12iaTODPJ*	0.1940	0.0532	0.1015	0.5407

We observe in Table 8 that, again, the order of the approaches for result merging in FedWeb 2012 do not agree with the one in FedWeb 2013. In terms of P@10 the best methods are different (boost vs relevance based, among the submitted ones). In terms of nDCG, the relevance based run is in both cases the best method, but the performance of the diversity approach is much lower than that of the boost one, and similar to the cluster run in FedWeb 2012; however, in FedWeb 2013 the diversity run outperforms the boost and cluster runs. This suggests that, when the resource ranking is not good (the performance of the hybrid method in resource selection is far from optimal), the diversification approach seems to help a little bit. On the other hand, the boosting method is highly dependent on the ranking of the resources, as we observe when a better resource selection method is used (BM25 desc in FedWeb 2013 or the hybrid run in FedWeb 2012).

3.5 Discussion

After participating in the Federated Search track, we have discovered that training techniques in an older dataset, even if its characteristics are very similar to the current one, does not guarantee a reproduction of the results obtained, making it very difficult to have reliable judgements about which techniques will perform better. We argue that this may be due to the content of the search engines changing from one collection to the other, but also because the queries, and in particular, its specificity (e.g., can it be answered by a general search engine or is it tailored to more focused, specialised engines?) may drastically change.

4. KNOWLEDGE BASE ACCELERATION

4.1 Track Description

In this section we describe our work in TREC Knowledge Base Acceleration (KBA) Cumulative Citation Recommendation (CCR), a task that aims at filtering a stream for documents that are citation-worthy for Wikipedia or Twitter entities of interest.

4.2 Motivation

Our participation in KBA 2013 was inspired by a desire to combine the best performing aspects of several approaches. In TREC-KBA 2012, we experimented with several approaches including string-matching and string-learning [2]. With string-matching, we represented the entities with rich features from a resource called Google Cross Lingual Dictionary (GCLD) which is a mapping (with probability distributions) from strings to concepts and vice versa. The string-learning approach learns the context (characters) around the mention of an entity in the provided relevance judgements and builds a profile of the entity from the characters.

The string-matching approach achieved good performance in general, but it was very good at recall in particular. The string-learning approach was very good at precision indicating that context around entity mentions is important for determining relevance of a document to an entity. We noted also high-performing approaches from TREC 2012 included an approach that used entity and related entity mentions [2, 20]. Finally, we came across studies that use multi-step approaches and a huge feature set for CCR [4, 5] that also achieve good performance.

One of the studies is called multi-step classification approach which compares two approaches: 2-step and 3-step. Both of them start with an initial step which filters the stream for potentially relevant documents. The 3-step approach next trains a classifier to separate *garbage* and *neutral* from *relevant* and *central*, and finally trains another classifier to separate *relevant* from *central*. The 2-step approach achieves a better performance than that of the 3-step approach.

The second of the studies is related to [5], but trains a Learning to Rank algorithm (LTR) instead of classification [4]. The classification and LTR approaches of [4, 5] shared the same set of 68 distinct features.

4.3 Method

We combine all the strengths in all the above approaches in an attempt to benefit from the strengths of each. Thus, we gathered features from the different approaches and added some new ones making a huge initial feature set. We reduced the feature set using different methods until we have few powerful subset which we ranked according to information gain. We applied the approach to the 2012 task and our performance was encouraging (both F-measure and SU being above 4.0). Encouraged by our performance on 2012 task, we applied the approach to the 2013 CCR task.

4.3.1 Features

The multi-step classification and LTR approaches used a set of 68 (5 document, 1 entity, 24 document-entity and 38 temporal) features (all numeric) [4, 5]. Document and entity features are computed from processing the documents and entities respectively. Document-entity features are computed by aggregating scores of strings for which a match has been found in a document. For example, if we consider the Personalised Page Rank (PPR) feature, for

each entity, there are 100 related entities each with its PPR score. When processing a document entity pair, if a document matches strings from the entity's pre-constructed PPR, we add up the scores and the sum becomes the PPR score for that document-entity pair. We take the 68 features as provided by the authors[8] and add others from [2, 20], described below and some of them modified to suit our approach.

4.3.2 Google's Cross Lingual Dictionary (GCLD)

This is a mapping of strings to Wikipedia concepts and vice versa [32]. The GCLD corpus computes two probabilities: (1) the probability with which a string is used as anchor text to a Wikipedia entity and (2) the probability that indicates the strength of co-reference of an anchor with respect to other anchors to a given Wikipedia entity. We use the product of both for each string.

4.3.3 PPR

For each entity, we computed a PPR score from a Wikipedia snapshot and we kept the top 100 entities along with the corresponding scores.

4.3.4 Surface Form (sForm)

For each Wikipedia entity, we gathered DBpedia name variants. These are redirects, labels and names.

4.3.5 Context (contxL, contxR)

From the WikiLink corpus [30], we collected all left and right contexts (2 sentences left and 2 sentences right) and generated n-grams between uni-grams and quadro-grams for each left and right context. Finally, we select the 5 most frequent n-grams for each context.

4.4 Classification

The 2-step approach that we use, following [5] and [4], consists of filtering followed by classification or learning. The first step filters the stream for documents that are potentially relevant using DBpedia name variants of the Wikipedia entities. The second step trains classification or learning to rank (LTR) algorithm. In both cases, we treat *central* as positive, and *garbage* and *neutral* as negative examples. However, *relevant* is excluded from the training stage, as these may introduce confusing examples for the classifiers.

We train J48 (CL-J48) and Random forest Model (CL-RF) decision tree classifiers, as implemented in Weka. Thus, we take the same settings as described in [5].

4.5 Result and Analysis

4.5.1 Feature analysis

Our Feature Analysis and selection was done using the 2012 datasets and relevance judgements. We compared our results with the state of the art studies on TREC-KBA CCR of 2012.

Figure 4 shows the performances (F) of the three algorithms against feature addition on the 2012 CCR task. The features are sorted from left to right, in descending order, in terms of information gain. The plot of Classification CL-RF is based on the average performance for 10 different random initialisations. The plus sign on a feature indicates that we incrementally add the feature into the feature set to the left of it.

From Figure 4, we see that the performances of the three algorithms increase with the addition of features to the initial feature set, reach maxima and then decrease. The increase and decrease are not uniform. However, we can see that the three algorithms reach their respective maxima within the first 13 features. This indicates that adding more features does not always improve performance. In fact, performance deteriorates with more features. We have taken the best F scores from the three plots and they are shown along with three baselines in Table 9. We have included two of the highly performing methods on 2012 CCR task as baselines. From classification, the 2-step approach's Random Forest is used as a baseline (MC-RF). The second is LTR's Random Forest (LTR-RF).

Table 9: KBA Track. Performances comparison of our approach (lower half) with baselines (upper half). Best scores are highlighted.

Method	F	SU
MC-RF	0.360	0.263
LTR-RF	**0.390**	**0.369**
CL-RF	**0.402**	**0.396**
LTR-RF	0.394	0.411
CL-J48	0.388	0.306

The scores in Table 9 show that our reduced feature set performs better than the baselines on both performance measures. The most informative features, as measured by information gain and contribution to performance, are: name variants (GCLD), similarity features (cos, jac, kl), related entities (PPR), context, position of entity mention in the document, and length of body text. These features can serve as baseline features for CCR task.

4.5.2 Results on TREC-KBA 2013's CCR

Encouraged by the results on TREC-KBA 2012's CCR task, we applied our reduced feature set and two classification algorithms (J48 and Random Forest) to the 2013 CCR task. We used three sets of features: all 26 features, features up-to FirstPosNorm (FPN) and Features up-to MentionsBody (MB). We also used three training datasets: 2012, 2013, and 2012-2013 combined relevance judgements. We generated 3 J48 runs using all features and the training sets and 9 RF runs using 3 feature sets and 3 training sets.

Results for the groups for which performance was above the median are shown in Table 10. System name format is algorithm_feature set_training dataset_13. For example RF_all_13_13 stands for Random Forest using all features, trained on 2013 and applied on 2013[9].

Table 10 shows our best performance according to micro average F and SU. The scores are obtained from the classification confidence scores and the classification label. We map the scores of irrelevant document-entity pair to (0, 500] and the scores of relevant to (500, 1000]. For vital classification, the highest score is on the Turing group. On all entities, micro-average-F is 0.247, and on Wikipedia entities, it is 0.290. On vital+useful, we have done well, achieving performance of 0.603 on all entities and 0.649 on Wikipedia only.

Our approach was very weak in Twitter entities achieving F-measure of 0.0. The low performance on Twitter entities is, of course, expected since almost all of the strong features we used did not apply to Twitter entities. For example, all the similarity (cos, kl, jac), GCLd, PPR, sform and context features were assigned 0

[8] http://krisztianbalog.com/files/resources/oair2013-kba/runs.zip

[9] On our run submissions we used RM to mean RF. Please replace every RM with RF

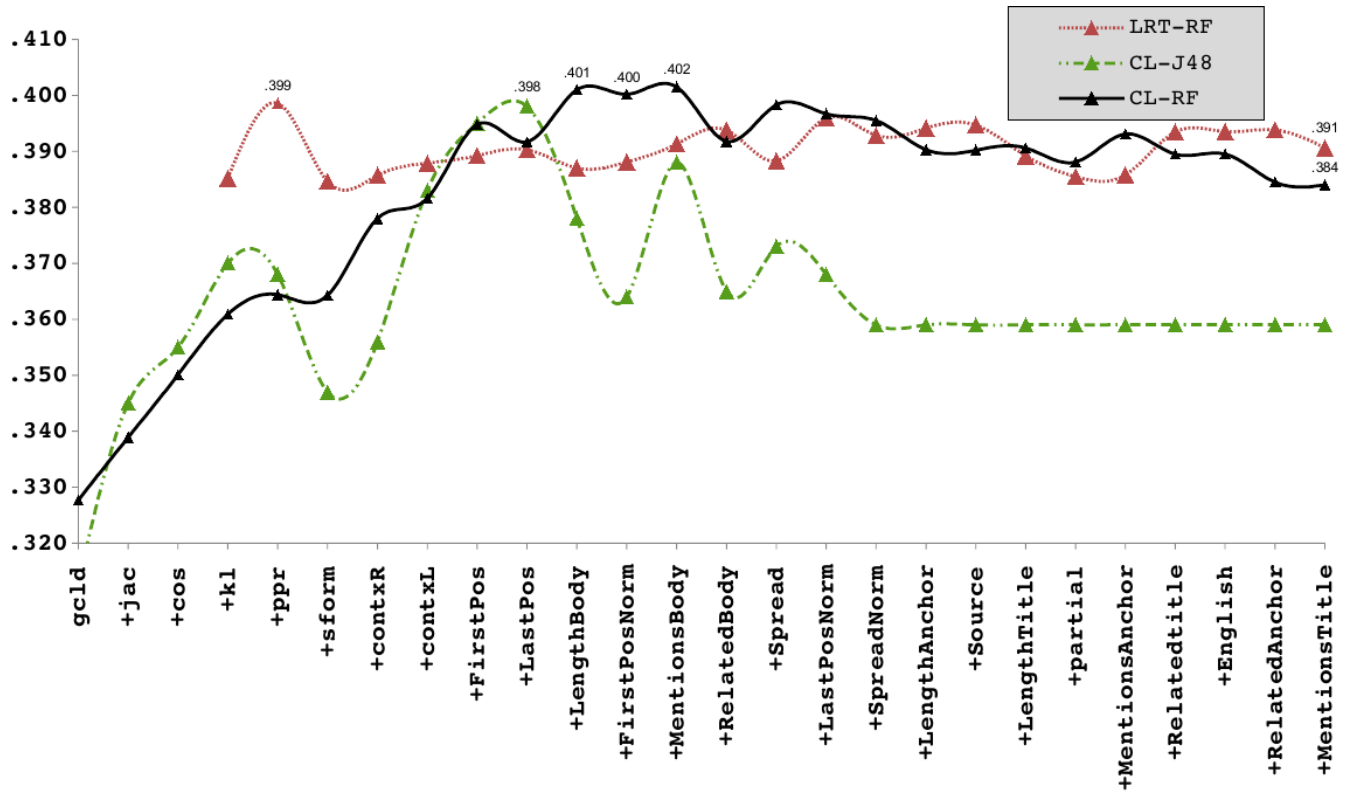

Figure 4: KBA Track. Performance (F) of classification and LTR algorithms against feature addition.

Table 10: KBA Track. System Performances on vital (upper half) and vital+useful (lower half).

system_id	F	SU	Group
RF _all_13_13	0.575	0.571	turing
RF _MB_13_13	0.388	0.404	fargo
RF _all_1213_13	0.353	0.370	hep
RF _FPN_1213_13	0.338	0.333	ocala
RF _FPN_12_13	0.298	0.395	bronfman
RF _MB_13_13	0.290	0.333	wikipedia
RF _FPN_13_13	0.279	0.422	danville
RF _MB_13_13	0.247	0.333	all-entities
RF _MB_13_13	0.241	0.341	hoboken
J48_13_13	0.232	0.333	screenwriters
RF _all_13_13	0.649	0.647	wikipedia
RF _MB_1213_13	0.603	0.602	all-entities

score. We also performed very poorly on the groups of startups, french, mining and comedians.

From algorithms, RF performs better in almost all cases. Regarding training dataset, we see that 2013 relevance judgements help train a better model. In many cases, training on 2012 data achieved 0.0 or very low performance. This is probably due to the fact that the CCR task has been changed from its 2012 definition.

Our performance on 2012 CCR task did not translate well to the 2013 CCR task. We suspect that this has to do with the change of the CCR task. We will need to do further analysis. However, we have achieved good results for some groups. We want to do further analysis of why we achieve better results for some groups and bad results for others.

5. AD-HOC WEB

The goal of our Web Track participation is to evaluate the improvement of our proximity model over its baseline. Our model has no specific mechanism for diversity or risk sensitivity, and therefore we do not consider our participation to be competitive to that of other participants, on the metrics that are used for the evaluation. We therefore only participated in the Web Track ad-hoc task. This section describes our participation.

5.1 Track description

The TREC Web Track ad-hoc task investigates the performance of systems that search a static set of documents using previously-unseen topics. The goal of the task is to return a ranking of the documents in the collection in order of decreasing probability of relevance.

5.2 Term Proximity

Ranking functions typically consider query terms as being independent, which simplifies the ranking problem. However, the user is often interested in the joint co-occurrence of query terms, forming a noun phrase or being otherwise related in the text. One feature that can be used to express the relatedness of co-occurring terms is their proximity in text. Intuitively, researchers have suspected that query terms that appear closer together in documents represent stronger evidence for relevance [8, 23, 24, 33, 37], however past research has shown this is not an easy feature to exploit.

5.3 Related Work

In the past, there has been much interest into automating query reformulation with respect to term dependency [6–10, 13–15, 19, 23, 24, 26, 29, 31, 33, 37]. Various ways have been proposed to promote documents in which query terms appear at closer proximity. Some approaches assign a score to each span of text that appears in a document containing multiple query terms [6, 8, 14, 24, 37], while others restrict their proximity model to using only term-pairs [26, 33]. The scope of proximity that is considered is often limited to a fixed word distance, for instance 5 terms by Rasolofo and Savoy [26] and 50 terms by Song et al. [31]. The score of the term-combinations found in a document is often added to a baseline that considers the terms to be independent [6, 24, 33, 37], while other models modify the relevance contribution of appearing terms [14, 31].

5.4 Baseline

Zhai and Lafferty [36] presented a function to rank documents according to the (negative) Kullback-Leibler (KL) divergence between a language model of query $\mathcal{Q}$ and a Dirichlet-smoothed language model of document $\mathcal{D}$. Each document $\mathcal{D}$ obtains a score for query $\mathcal{Q}$ using Equation 5, where t is a term in $\mathcal{Q}$, $P\left(t|\mathbb{C}\right)$ is the likelihood of term t appearing in the corpus, $c\left(t,\mathcal{D}\right)$ is the term frequency of term t in document $\mathcal{D}$, $P\left(\mathcal{D}\right)$ is a term independent document prior (Equation 6), $|\mathcal{Q}|$ is the number of terms in $\mathcal{Q}$, and μ is the Dirichlet smoothing parameter. We refer to this function as KLD in this paper.

$$KLD\left(\mathcal{Q},\mathcal{D}\right) = P\left(\mathcal{D}\right) + \sum_{t\in\mathcal{Q}} \log\left(1 + \frac{c\left(t,\mathcal{D}\right)}{\mu \cdot P\left(t|\mathbb{C}\right)}\right) \quad (5)$$

$$P\left(\mathcal{D}\right) = |\mathcal{Q}| \cdot \log\frac{\mu}{\mu + |\mathcal{D}|} \quad (6)$$

In this study, we used the KLD function as the base function for ranking, expanding it with a proximity model to promote documents based on the distance between co-occurring query terms.

5.5 Proximity Model

Existing work indicates that documents that contain query terms at closer proximity are more likely to be relevant than documents that contain the same terms over longer distance. Of many attempts, only a few appear successful in exploiting this feature in a ranking function [24, 33]. Both methods emphasise near co-occurrences; Metzler and Croft [24] used only co-occurrences within a very small window and Tao and Zhai [33] used only the smallest distance between any two different query terms in a document. The use of these near occurrences alone is shown to give significant improvement over a non-proximity baseline. However, the need to exclude co-occurrences over longer distance reveals the need to properly balance the relevance estimation of term co-occurrences with respect to their distance in text.

We hypothesise that proximity between co-occurring query terms can still be useful over longer distance, for instance to decide which document is most likely to be relevant between two documents that are the same in other respects. A model that uses all available proximity information can outperform a model that uses less information, if the estimation of relevance for co-occurring terms over distance is sufficiently accurate. For this, we analysed the likelihood that term co-occurrences appear in a relevant document, by counting all term co-occurrences in the TREC 1-8 ad-hoc queries that according the qrels either appear in relevant or irrelevant documents. Using these statistics we estimate the likelihood that a term co-occurrence given a certain distance appears in a relevant document. This likelihood is inversely proportional to the distance of the terms in text. We also observed that adjacently appearing terms are close to twice as likely to appear in relevant documents than terms that most likely appear independently based on their span. Given that in the baseline function (Equation 5) each occurring independent term is scored as a count of 1, we designed a function that estimates the relevance of a term co-occurrence to be 1 for adjacent terms, decaying with 1 over distance and approaching 0 at infinity. In Equation 7, $\mathcal{S}$ is a set consisting of two or more query terms, $\mathcal{O}$ is an occurrence of $\mathcal{S}$ in a document $\mathcal{D}$, and $span\left(\mathcal{O}\right)$ is the number of word positions of a text in $\mathcal{D}$ that starts with a term from $\mathcal{S}$, ends with a term from $\mathcal{S}$, and contains all terms from $\mathcal{S}$.

$$R\left(\mathcal{O},\mathcal{S}\right) = \frac{|\mathcal{S}| - 1}{span\left(\mathcal{O}\right) - 1} \quad (7)$$

Metzler and Croft [24] argued that terms that appear adjacent in the query are more important candidates than terms that are separated. However, in our pre-study we found that distinguishing between the weights of adjacent and non-adjacent query terms hardly improved results over simply using all possible term combinations using the same weight.

Our ranking function in Equation 8 combines the KLD baseline with the additional score for the term proximities. For balancing the respective weight of the baseline and the proximity models, we have to compensate for the fact that the number of term combinations grow at a rate of $2^{|\mathcal{Q}|} - |\mathcal{Q}| - 1$ possible term combinations. In practice, the growth ratio of the weight of proximity model with respect to the size of the query does not grow exponentially, but increases with a ratio of $|\mathcal{Q}|$, which we determined empirically. This less than exponential growth we observed can partly be explained by combinations of many terms being very rare. Term co-occurrences are scored as separate evidence over the unigrams they contain.

In Equation 9, the proximity model PM uses all combinations of two or more query terms, defined as an element of the powerset over $\mathcal{Q}$ with a cardinality greater than 1. Each term combination $\mathcal{S}$ is scored as the sum of the KLD function over all terms contained, replacing the original count of the unigram in the document with the estimated relevance of the term co-occurrence (Equation 10). In Equation 10, we estimate the relevance of each term combination $\mathcal{S}$ by adding up the relevance estimation of each occurrence $\mathcal{O}$ of that combination in document $\mathcal{D}$ using Equation 7.

$$score\left(\mathcal{Q},\mathcal{D}\right) = \frac{1}{|\mathcal{Q}|} \cdot PM\left(\mathcal{Q},\mathcal{D}\right) + KLD\left(\mathcal{Q},\mathcal{D}\right) \quad (8)$$

$$PM\left(\mathcal{Q},\mathcal{D}\right) = \sum_{\mathcal{S}\in\mathcal{P}_{\geq 1}\left(\mathcal{Q}\right)} \sum_{t\in\mathcal{S}} \log\left(1 + \frac{R\left(\mathcal{S},\mathcal{D}\right)}{\mu \cdot P\left(t|\mathbb{C}\right)}\right) \quad (9)$$

$$R\left(\mathcal{S},\mathcal{D}\right) = \sum_{\mathcal{O}} R\left(\mathcal{O},\mathcal{S}\right) \quad (10)$$

5.6 Stop words

In general, retrieval performance improves when stop words are removed from the query. The rationale for this is that stop words rarely occur more often in relevant documents, not even if they convey part of the information need by being part of a noun phrase, phrase of speech, describing a relation or complementing another term. Meaningful stop words may however occur closer to other query terms in relevant documents, making them potentially useful in proximity models.

The proximity model (Equation 9) uses the power set $\mathcal{P}_{\geq 1}\left(\mathcal{Q}\right)$, where $\mathcal{Q}$ is the query from which stop words are removed. For this variant, we can use the same proximity model by replacing this power set with a set that contains combinations with stop words.

To avoid irrelevant combinations with stop words from being used, we use a simple heuristic: a combination of query terms is only valid when all stop words used are used in combination with all words that connect them in the query to non stop words or the query boundary. For example, "The Beatles on a zebra crossing" would generate besides "Beatles zebra", "zebra crossing", "Beatles crossing" and "Beatles zebra crossing", the following combinations containing stop words: "The Beatles", "The Beatles on a zebra", "The Beatles zebra", "The Beatles zebra crossing", "The Beatles crossing", "Beatles on a zebra", "Beatles on a zebra crossing", and "The Beatles on a zebra crossing".

To test this hypothesis, we will run a model that contains all stop words as a variant of the proximity model.

5.7 Results

For this study, we indexed ClueWeb12 using a Porter-2 stemmer, lowercasing terms and storing term-document positions in the posting lists. Stop words were included in the index, but removed from the queries for the KLD baseline and the proximity model CPE, but used for the proximity model with stop words CPS as described.

Table 11: Ad-hoc Web Track. Comparison of three runs and the median of all Web Track runs using ERR. Results marked with † are significant over KLD and results marked with ‡ are significant over CPE, tested using 1-tailed paired Student's T-test, α = 0.05.

runid	ERR-IA@5	ERR-IA@10	ERR-IA@20
cwiwt13kld	0.3032	0.3272	0.3363
cwiwt13cpe	0.3912 †	0.4081 †	0.4177 †
cwiwt13cps	0.4555 †‡	0.4726 †‡	0.4803 †‡
median			0.4739

Table 12: Ad-hoc Web Track. Comparison of three runs and the median of all Web Track runs using α-nDCG. Results marked with † are significant over KLD and results marked with ‡ are significant over CPE, tested using 1-tailed paired Student's T-test, α = 0.05.

runid	α-nDCG@5	α-nDCG@10	α-nDCG@20
cwiwt13kld	0.3456	0.3970	0.4284
cwiwt13cpe	0.4342 †	0.4713 †	0.5033 †
cwiwt13cps	0.4920 †	0.5309 †‡	0.5572 †‡
median			0.5745

Table 11 and 12 compare the results over the metrics that emphasise diversification. These results show both proximity models to consistently improve over the KLD baseline. The proximity model that uses stop words performs significantly better than the proximity model without stop words. CPS scores close to the median of all TREC participants, but should not be considered a real competitor having no mechanism to handle diversity.

Table 13 compares the results over ERR@20 and nDCG@20, and Table 14 and Table 15 show the relative improvement in percentage of the proximity models over KLD and the track median. On these non-diversified metrics the proximity models outperform both baselines.

Table 13: Ad-hoc Web Track. Comparison of three runs and the median of all Web Track runs using nDCG@20 and ERR@20. Results marked with † are significant over KLD and results marked with ‡ are significant over CPE, tested using 1-tailed paired Student's T-test, α = 0.05.

runid	nDCG@20	ERR@20
cwiwt13kld	0.1648	0.0850
cwiwt13cpe	0.1910	0.1090
cwiwt13cps	0.2181 †	0.1285 †‡
median	0.1739	0.0980

Table 14: Ad-hoc Web Track. Relative improvement of CPE and CPS over KLD and the track median, based on the nDCG@20results.

runid	nDCG@20	ERR@20
cwiwt13cpe	+15.9%	+9.8%
cwiwt13cps	+32.4%	+25.5%

Table 15: Ad-hoc Web Track. Relative improvement of CPE and CPS over KLD and the track median, based on the ERR@20results.

runid	cwiwt13kld	median
cwiwt13cpe	+28.2%	+11.1%
cwiwt13cps	+51.1%	+31.0%

5.8 Discussion

We participated in the Web Track ad-hoc task to evaluate a proximity model that uses the maximum available information. Our first hypothesis is that using the distance between query terms in a document is useful over longer distance if a sufficiently accurate relevance estimation is used. Although the results by the CPE model are encouraging, we will need additional experiments to show that using occurrences over longer distance is indeed beneficial. Our second hypothesis is that retrieval performance can be improved by including term combinations with stop words in the proximity model. The results show that the proximity model that uses stop words significantly improved results over the KLD baseline and also consistently outperformed the model that uses no stop words.

Acknowledgements

This research was supported by the Netherlands Organization for Scientific Research (NWO project #640.005.001) and under COMMIT project Infiniti. Part of this work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme, funded by European Comission FP7 grant agreement no.246016.

6. REFERENCES

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM*, pages 5–14, 2009.

[2] S. Araujo, G. Gebremeskel, J. He, C. Bosscarino, and A. de Vries. CWI at TREC 2012, KBA track and session track. In *TREC*, 2012.

[3] J. Attenberg and F. J. Provost. Why label when you can search? Alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *KDD*, pages 423–432, 2010.

[4] K. Balog and H. Ramampiaro. Cumulative citation recommendation: Classification vs. ranking. In *SIGIR*, pages 941–944, 2013.

[5] K. Balog, H. Ramampiaro, N. Takhirov, and K. Nørvåg. Multi-step classification approaches to cumulative citation recommendation. In *OAIR*, pages 121–128, 2013.

[6] M. Bendersky and W. B. Croft. Modeling higher-order term dependencies in information retrieval using query hypergraphs. In *SIGIR*, pages 941–950, 2012.

[7] S. Büttcher, C. L. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *SIGIR*, pages 621–622, 2006.

[8] C. L. Clarke, G. V. Cormack, and E. A. Tudhope. Relevance ranking for one to three term queries. *Information Processing & Management*, 36(2):291–311, 2000.

[9] W. B. Croft, H. R. Turtle, and D. D. Lewis. The use of phrases and structured queries in information retrieval. In *SIGIR*, pages 32–45, 1991.

[10] R. Cummins and C. O'Riordan. Learning in a pairwise term-term proximity framework for information retrieval. In *SIGIR*, pages 251–258, 2009.

[11] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 107–144. Springer, Boston, MA, 2011.

[12] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.

[13] J. Fagan. Automatic phrase indexing for document retrieval. In *SIGIR*, pages 91–101, 1987.

[14] J. Gao, J.-Y. Nie, G. Wu, and G. Cao. Dependence language model for information retrieval. In *SIGIR*, pages 170–177, 2004.

[15] B. He, J. X. Huang, and X. Zhou. Modeling term proximity for probabilistic information retrieval models. *Information Sciences*, 181(14):3017–3031, 2011.

[16] J. He, E. Meij, and M. de Rijke. Result diversification based on query-specific cluster ranking. *Journal of the American Society for Information Science and Technology*, 62(3):550–571, 2011.

[17] J. He, V. Hollink, and A. de Vries. Combining implicit and explicit topic representations for result diversification. In *SIGIR*, pages 851–860, 2012.

[18] G. Hubert and G. Cabanac. IRIT at TREC 2012 contextual suggestion track. In *TREC*, 2012.

[19] E. M. Keen. The use of term position devices in ranked output experiments. *Journal of Documentation*, 47(1):1–22, 1991.

[20] X. Liu and H. Fang. Leveraging related entities for knowledge base acceleration. In *Web-KR*, 2013.

[21] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer, Boston, MA, 2011.

[22] H. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, pages 159–165, 1958.

[23] Y. Lv and C. Zhai. Positional language models for information retrieval. In *SIGIR*, pages 299–306, 2009.

[24] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *SIGIR*, pages 472–479, 2005.

[25] D. Nguyen, T. Demeester, D. Trieschnigg, and D. Hiemstra. Federated search in the wild: the combined power of over a hundred search engines. In *CIKM*, pages 1874–1878, 2012.

[26] Y. Rasolofo and J. Savoy. Term proximity scoring for keyword-based retrieval systems. In *Advances in Information Retrieval*, pages 207–218, 2003.

[27] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186, 1994.

[28] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11): 613–620, 1975.

[29] L. Shi and J.-Y. Nie. Using various term dependencies according to their utilities. In *CIKM*, pages 1493–1496, 2010.

[30] S. Singh, A. Subramanya, F. Pereira, and A. McCallum. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst, 2012.

[31] R. Song, M. J. Taylor, J.-R. Wen, H.-W. Hon, and Y. Yu. Viewing term proximity from a different perspective. In *Advances in Information Retrieval*, pages 346–357, 2008.

[32] V. I. Spitkovsky and A. X. Chang. A cross-lingual dictionary for English Wikipedia concepts. In *LREC*, pages 3168–3175, 2012.

[33] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *SIGIR*, pages 295–302, 2007.

[34] G. M. Weiss. Learning with rare cases and small disjuncts. In *ICML*, pages 558–565, 1995.

[35] P. Yang and H. Fang. Opinion-based user profile modeling for contextual suggestions. In *ICTIR*, page 18, 2013.

[36] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, pages 334–342, 2001.

[37] J. Zhao and Y. Yun. A proximity language model for information retrieval. In *SIGIR*, pages 291–298, 2009.