Session Track Overview*

Evangelos Kanoulas[†] Ben Carterette[‡] Paul Clough[§] Mark Sanderson[¶]

October 25, 2010

Abstract

Research in Information Retrieval has traditionally focused on serving the best results for a single query. In practice however users often enter queries in sessions of reformulations. The Sessions Track at TREC 2010 implements an initial experiment to evaluate the effectiveness of retrieval systems over single query reformulations.

1 Introduction

Research in Information Retrieval has traditionally focused on serving the best results for a query.But users often begin an interaction with a search engine with a sufficiently ill-specified query that they will need to reformulate before they find what they are looking for: early studies on web search query logs showed that half of all Web users reformulated their initial query: 52% of the users in 1997 Excite data set, 45% of the users in the 2001 Excite dataset [**?**]. A search engine may be able to better serve a user not by ranking the most relevant results to each query in the sequence, but by ranking results that help "point the way" to what the user is really looking for, or by complementing results from previous queries in the sequence with new results, or in other currently-unanticipated ways.

The standard evaluation paradigm of controlled laboratory experiments is unable to assess the effectiveness of retrieval systems to an actual user experience of querying with reformulations. On the other hand, interactive evaluation is both noisy due to the high degrees of freedom of user interactions, and expensive due to its low reusability and need for many test subjects. The TREC 2010 Session Track is an attempt to evaluate the simplest form of user interaction with a retrieval engine: a single query reformulation.

2 Evaluation Tasks

We call a sequence of reformulations in service of satisfying an information need a "session", and thes goal of this track are: (**G1**) to test whether systems can improve their performance for a given query by using a previous query, and (**G2**) to evaluate system performance over an entire query session instead of a single query.

For this first year, we limited the focus of the track to sessions of two queries, and further limited the focus to particular types of sessions (described in Section 3.2). This is partly for pragmatic reasons regarding

*Notebook Version

[†]Information School, University of Sheffield, Sheffield, UK
[‡]Department of Computer & Information Sciences, University of Delaware, Newark, DE, USA
[§]Information School, University of Sheffield, Sheffield, UK
[¶]Department of Computer Science & Information Technology, RMIT University, Melbourne, Australia

the difficulty of obtaining session data, and partly for reasons of experimental design and analysis: allowing longer sessions introduces many more degrees of freedom, requiring more data from which to base conclusions.

A set of 150 query pairs (initial query, query reformulation) was provided to participants by NIST. For each such pair the participants submitted 3 (three) ranked lists of documents for three experimental conditions,

1. one over the initial query (**RL1**)
2. one over the query reformulation, ignoring the initial query (**RL2**)
3. one over the query reformulation taking into consideration the initial query (**RL3**)

By using the ranked lists (RL2) and (RL3) we evaluated the ability of systems to utilize prior history (G1). By using the returned ranked lists (RL1) and (RL3) we evaluate the quality of ranking function over the entire session (G2). Note that this was not be an interactive track. Query reformulations were provided by NIST along with the initial queries. Further note that when retrieving results for (RL3) the only extra information about the user's intent is the initial query. This was a single-phase track, with no feedback provided by the assessors.

3 Test Collection

3.1 Corpus

The track used the ClueWeb09 collection. The full collection consists of roughly 1 billion web pages, comprising approximately 25TB of uncompressed data (5TB compressed) in multiple languages. The dataset was crawled from the Web during January and February 2009. Participants were encouraged to use the entire collection, however submissions over the smaller "Category B" collection of 50 million documents were accepted. Note that Category B submissions was evaluated as if they were Category A submissions.

3.2 Queries and Reformulations

There is a large volume of research regarding query reformulations which follows two lines of work: a descriptive line that analyzes query logs and identifies a taxonomy of query reformulations based on certain user actions over the initial query (e.g. [**?**, **?**]) and a predictive line that trains different models over query logs to predict good query reformulations (e.g. [**?**, **?**, **?**, **?**]). Analyses of query logs have shown a number of different types of query reformulations with three of them being consistent across different studies (e.g. [**?**, **?**]):

Specifications: the user enters a query, realizes the results are too broad or that they wanted a more detailed level of information, and reformulates a more specific query.

Drifting/Parallel Reformulation: the user entered a query, then reformulated to another query with the same level of specification but moved to a different aspect or facet of their information need.

Generalizations: the user enters a query, realizes that the results are too narrow or that they wanted a wider range of information, and reformulated a more general query.

In the absence of query logs, Dang and Croft [**?**] simulated query reformulations by using anchor text, which is readily available. In the Session Track we used a different approach. To construct the query pairs (initial query, query reformulation) we started with the TREC 2009 and 2010 Web Track diversity topics. This collection consists of topics that have a "main theme" and a series of "aspects" or "sub-topics". The Web Track queries were sampled from the query log of a commercial search engine and the sub-topics were constructed by a clustering algorithm [**?**] run over these queries aggregating query reformulations occurring in

the same session. We used the aspect and main theme of these collection topics in a variety of combinations to provide a simulation of an initial and second query. An example of part of a 2009 Web track query is shown below.

```
<topic number="4" type="faceted">
  <query>toilet</query>
  <description> Find information on buying, installing, and repairing
  toilets.
  </description>
  <subtopic number="1" type="inf">
    What different kinds of toilets exist, and how do they differ?
  </subtopic>
  <subtopic number="2" type="inf">
    I'm looking for companies that manufacture residential toilets.
  </subtopic>
  <subtopic number="3" type="inf">
    Where can I buy parts for American Standard toilets?
  </subtopic>
  <subtopic number="4" type="inf">
    How do I fix a toilet that isn't working properly?
  </subtopic>
  <subtopic number="5" type="inf">
    What companies manufacture bidets?
  </subtopic>
  <subtopic number="6" type="inf">
    I'm looking for a Kohler wall-hung toilet.  Where can I buy one?
  </subtopic>
</topic>
```

To construct **specification** reformulations (1) used the Web Track ¡query¿ section as the initial query, (2) we selected one of the subtopics and used the ¡subtopic¿ section as the description of the actual information need, and (3) we then extract keywords out of the subtopic description and used these keywords as the query reformulation. For instance, in the example above we used the Web Track query "toilet" as the first query, we selected one of the subtopics as the information need ("I'm looking for a Kohler wall-hung toilet. Where can I buy one?") and we extract the keyword "Kohler" and used it as the second query (query reformulation). Essentially, this example simulates a user that is actually looking for a Kohler wall-hung toilet but he poses a more general query to the search engine ("toilet"). Given that "toilet" is a quite general term the user reformulates his query to "Kohler" to find web pages closer to his information need.

```
<topic number="1" reformtype="specification" source="webtrack2009">
  <query>toilet</query>
  <reformulation>Kohler</reformulation>
  <description>I'm looking for a Kohler wall-hung toilet.
               Where can I buy one?</description>
</topic>
```

To construct **drifting** reformulations (1) we selected two of the subtopics and used the ¡subtopic¿ sections as the description of the two information needs, and (2) we then extract keywords out of the subtopic descriptions and used these keywords as the first query and the query reformulation. For instance, in the example above we selected "Where can I buy parts for American Standard toilets?" and "I'm looking for a Kohler wall-hung toilet. Where can I buy one?" as the two information needs. Then we extracted the

keywords “American Standard” and “Kohler” and used them as the initial query and the query reformulation. Essentially this reformulation simulates a user that first wants to buy some toilet parts from American Standard but while he is browsing the results of the query or possibly other web pages he decides that he also wants to purchase Kohler wall-hungs and thus there is a slight drifting in his information need.

```
<topic number="2" reformtype="drifting" source="webtrack2009">
  <query>American Standard</query>
  <description>Where can I buy parts for American Standard toilets?</description
  <reformulation>Kohler</reformulation>
  <rdescription>I'm looking for a Kohler wall-hung toilet.
                Where can I buy one?</rdescription>
</topic>
```

Finally, to construct **generalization** reformulations we followed a slightly more complicated process. The reason is that the Web Track queries include queries that are under-specified and then subtopics that are the exact specification of different information needs but it does not include over-specified or mis-specified queries that could lead the user to generalize his initial query.

Thus in the first method to construct generalization reformulation (1) we selected one of the subtopics and we extracted as many keywords as possible to construct an over specified query, e.g. (we selected “What different kinds of toilets exist, and how do they differ?” and we extracted the keywords “different kinds of toilets” which seems to be a lexical over-specification), (2) we used a subset of these keywords to generalize the initial query (e.g. “toilet”). Essentially this reformulation simulates a user that first wanted to find what different kinds of toilets exist, and how do they differ but he lexically over-specified his need, the returned results were poor and thus reformulated his initial query to a more general one.

```
<topic number="3" reformtype="generalization" source="webtrack2009">
  <query>different kinds of toilets</query>
  <reformulation>toilets</reformulation>
  <description> What different kinds of toilets exist,
                and how do they differ?</description>
</topic>
```

The second method used was to essentially consider that a user mis-specifies his need to something very narrow and then he generalizes. In this case (1) we selected one of the subtopics or query description from the Web Track topics as the information need, (2) extracted keywords from a different subtopic that seemed related but essentially it was a mis-specification of something very narrow, and (3) extracted keywords from the subtopic used as information need.

```
<topic number="4" reformtype="generalization" source="webtrack2009">
  <query>American Standard toilet</query>
  <reformulation>toilet</reformulation>
  <description>Find information on buying, installing,
               and repairing toilets.</description>
</topic>
```

Furthermore, given that not all queries in the Web Track proved appropriate to construct query reformulations we used some of the Million Query 2009 (MQ09) queries and built query reformualations and information needs from scratch. The MQ09 queries was used as the initial queries. Often they were submitted to a web search engine whose results were used as an assistance to create information needs. Queries from the “Related Search” section of Bing were also used as an assistance to construct query reformulations. Often,

Bauhaus University Weimar	Gale, Cengage Learning
Hungarian Academy of Science	RMIT University
The University of Melbourne	University of Amsterdam
University of Arkansas at Little Rock	University of Delaware
University of Essex	University of Lugano

Table 1: Groups participating in the 2010 Sessions Track.

if the queries from the "Related Search" were not good examples one of them was selected, re-submitted to Bing and the new "Related Search" section was used for assistance. Examples of such queries can be viewed bellow.

```
<topic number="47" reformtype="specification" source="mqtrack2009">
  <query>cuttle fish bone</query>
  <reformulation>cuttlebone casting</reformulation>
  <description>Find information about the cuttlebone metal casting
               technique and how can it be used to create jewelry?</description>
</topic>

<topic number="38" reformtype="drifting" source="mqtrack2009">
  <query>sun spot activity</query>
  <description>What is sunspot activity and what causes it?</description>
  <reformulation>sun spot earthquake</reformulation>
  <rdescription>Can sunspot activity be used to predict earthquakes?
                Are there any studies?</rdescription>
</topic>

<topic number="41" reformtype="generalization" source="mqtrack2009">
  <query>history crossword</query>
  <reformulation>history games</reformulation>
  <description>Find printable history games to be used in a
               history class.</description>
</topic>
```

A set of 150 (initial query, query reformulation) pairs[1] were provided to participants, 52 of which were specification reformulations, 50 were drifting reformulations and 48 were generalization reformulations.

4 Submissions

Sites were permitted to submit up to three runs. Each submitted run includes three separate ranked result lists for all 150 topics. Files were named "runTag.RLn", where "runTag" is a unique identifier for the site and the particular submission, and "RLn" is RL1, RL2, or RL3, depending on the experimental condition.

The track received 27 runs from the 10 groups listed in Table 1. Seven sites submitted three runs and three sites submitted two runs.

Section A summarizes the methods used by each of the participating sites. For further details on the techniques used refer to the individual groups reports for the Session Track.

[1]Pairs, topics and sessions are used interchangeably in this document.

5 Session Evaluation

5.1 Relevance Judgments

Judging was done by assessors at NIST. The top 10 documents of all the ranked lists submitted by all participants (i.e. RL1, RL2, and RL3) were pooled together for each one of the topics (depth-10 pooling). For each topic the NIST assessors judged documents with respect to one or two information needs simultaneously. In the case of specification and generalization reformulations, both the initial query and its reformulation were assumed to represent the same information need, and assessors judged all documents against that need. In the case of drifting reformulations the query reformulation represented an information need different from (but related to) the information need of the initial query, and thus judgments were made with respect to both needs. (Note that the information needs were not provided to participants.)

Due to limited resources, topics assigned to each NIST assessor were prioritized by the size of the depth-10 pool – the smaller the pool size the higher the priority of the topic – and NIST assessors rotated between the three types of query reformulations preserving this order. The judging process lasted approximately two weeks and judgments were provided for 136 out of the 150 sessions. The IDs of the topics that were not judged are the following: 24, 30, 35, 36, 40, 58, 70, 100, 114, 118, 120, 126, 130, and 136. The judged 136 topics include 47 specification, 47 drifting and 42 generalization reformulation types.

A total of 33,121 documents were judged, with 11,525 documents being judged with respect to two information needs, resulting in 44,646 relevance judgments. Of the 44,646 judgments, 2,958 (6.6%) were highly relevant, 4,798 (10.7%) were relevant, and 36,890 (82.7%) non-relevant.

The specification topics had an average of 15.5 highly relevant documents, 23.6 relevant, and 201 non-relevant. The generalization topics had more relevant material, with an average 23.1 highly relevant documents, 40.5 relevant, and 181 non-relevant. The drifting topics had an average of 13.8 highly relevant, 20.1 relevant, and 211.9 non-relevant documents for the initial information need, and 12.8, 22.1, and 210.1 respectively for the reformulated information need.

The format of the qrels file generated is

```
topicID 0 docID rel1.rel2
```

i.e. the typical qrels format except for the `rel` field that has been replaced by `rel.rel`. For generalization and specification topics, the value on the left of the decimal is the relevance judgment, and the right value is always -1. For drifting topics, the values are for the two information needs respectively.

5.2 Normalized session DCG

Järvelin et al. [**?**] extended the nDCG metric into a session metric that takes into account multiple *interactive* queries. The metric discounts documents that appear lower in the ranked list for a given query and further discounts documents that require query reformulations to be found. In a sense the new model on which the session-based nDCG (nsDCG) is defined incorporates a cost for reformulating a query.

The session-based nDCG is defined as follows: for each query in a series of reformulations, one can compute DCG, as defined above, in isolation of all other queries in the series, i.e. for a query q,

$$DCG@k(q) = \sum_{i=1}^{k} \frac{rel(i,q)}{(1 + log_b(i))}$$

where b is the base of the logarithm of the discount function and controls the persistence or patience of the user to move down the ranked list of documents for a given query. Each query in this series of reformulations

is penalized based on a similar discount as a function of the position q of the query in the series. Based on that the session DCG (sDCG) for a query in position q is defined as,

$$sDCG@k(q) = \frac{1}{(1 + log_{bq}(q))} * DCG@k(q)$$

where bq is the logarithm base for the query discount. The session DCG can be written then as,

$$\begin{aligned} sDCG@k(q) &= \frac{1}{(1 + log_{bq}(q))} \sum_{i=1}^{Z} \frac{rel(i,q)}{(1 + log_b(i))} \\ &= \sum_{i=1}^{Z} \frac{rel(i,q)}{(1 + log_{bq}(q))(1 + log_b(i))} \end{aligned}$$

As with the standard formulation of DCG, we can compute an "ideal" score based on an optimal ranking of documents in decreasing order of relevance to the query, then normalize DCG by that ideal score.

5.3 Evaluation metrics for the Session Track

We have evaluated the submitted runs by three metrics, (a) nsDCG@10 as described above, (b) a variant that penalizes duplicates in the second ranking (nsDCG_dupes@10), and (c) nDCG10 of each submitted ranking. The first two evaluate the entire session and thus for each one of the two metrics we compute an evaluation score for RL1→RL2 (nsDCG@10.RL12) and an evaluation score for RL1→RL3 (nsDCG@10.RL13).

nsDCG@10 for RL1→RL3 is the official metric for (**G2**). Comparing the nsDCG@10 or nsDCG_dupes@10 scores for RL1→RL2 and RL1→RL3, or the nDCG@10 scores for RL2 and RL3 we evaluate how well systems performed with respect to (**G1**).

The specific instantiation of nsDCG10 used is :

$$\sum_{r=1}^{10} \frac{2^{rel(r,RL1)} - 1}{(log_2(r+1)) * (log_4(1+3))} + \sum_{r=1}^{10} \frac{2^{rel(r,RL2/RL3)} - 1}{(log_2(r+10+1)) * (log_4(2+3))}$$

where rel(r,RL1) is the relevance of the document in rank r in RL1 and rel(r,RL2/RL3) is the relevance of the document at rank r in RL2 (or RL3).

nsDCG_dupes@10 is similar to the nsDCG@10 except that duplicate documents in RL2@[1..10] and RL3@[1..10] with respect to RL1@[1..10] (i.e. documents that appear in the top 10 ranks of RL2 and RL3 that have previously appeared in the top 10 ranks of RL1) are considered non-relevant. Note that in the case of drifting reformulations there are no duplicate documents since the reformulated query represents a different information need that the one captured by the initial query and thus the two metrics, nsDCG and nsDCG_dupes are exactly the same.

Further note that in the case of specification and generalization the ideal sDCG differs between the two metrics. The ideal sDCG@10 for snDCG@10 is computed by concatenating the top 10 components of the idealDCG vector twice with each repeated result discounted according to the sDCG formula. Essentially, the ideal DCG for each rank is the DCG corresponding to the optimal ranking of documents. Since, duplicate documents are considered relevant by nsDCG, the optimal ranking is the same both for the initial query and its reformulation. On the other hand the idealsDCG for snDCG_dupes@10 is computed by concatenating the idealDCG[1..10] with the idealDCG[11..20] and discounting appropriately. Since duplicate documents are considered non-relevant in this case the optimal ranking is the one in which the best 10 documents appear in ranks 1 to 10 in RL1 and the second best 10 documents appear in ranks 1 to 10 in RL2/RL3.

The reason for computing nsDCG_dupes was the fact that one way of utilizing the initial query to produce a better ranking for the query reformulation would be to remove from RL3 documents that appeared in RL1.

The snDCG metric does not award novelty in RL3. On the contrary, since retrieval systems that remove duplicates need to find more relevant documents to populate RL3[1..10], nsDCG actually penalizes novelty.

The instantiation of the nDCG@10 used was,

$$\sum_{r=1}^{10} \frac{2^{rel(r,RL1/RL2/RL3)} - 1}{(log_2(r+1))}$$

for the three ranked lists RL1, RL2 and RL3 in isolation.

6 Evaluation Results

In the sections below we present the results of the evaluation for the two goals of the track. We beginning with the overall session evaluation, i.e. the goal (**G2**), which measures how well retrieval systems performed over the entire session. Then we present the evaluation of the (**G1**) goal of the track, i.e. the ability of retrieval systems to utilize past user queries to improve the results over the current query.

6.1 Session Evaluation: G2

Figure 1 shows the ranking of systems by nsDCG.RL13 and nsDCG_dupes.RL13, i.e. the nsDCG for the session RL1→RL3, grouped by participating sites. Table 2 reports the snDCG@10 scores for the session RL1→RL3 over all sessions and the specification, generalization and drifting sessions in isolation. Runs are sorted by snDCG@10.RL13 for all sessions. Some of the runs whose ranking over particular reformulation types appears to significantly diverge from the over all sessions ranking are marked by boldface. As it can be observed, overall, systems appear to perform better over the generalization and drifting sessions than the specification ones. The Kendall's τ correlations between the rankings for all the sessions, the specification, the generalization and the drifting sessions appears in the table below. It is clear that there is a strong "System – Reformulation Type" interaction effect.

	Kendall's τ		
	Specification	Generalization	Drifting
All sessions	0.8461538	0.8176638	0.8176638
Specification		0.7321937	0.6638177
Generalization			0.6695157

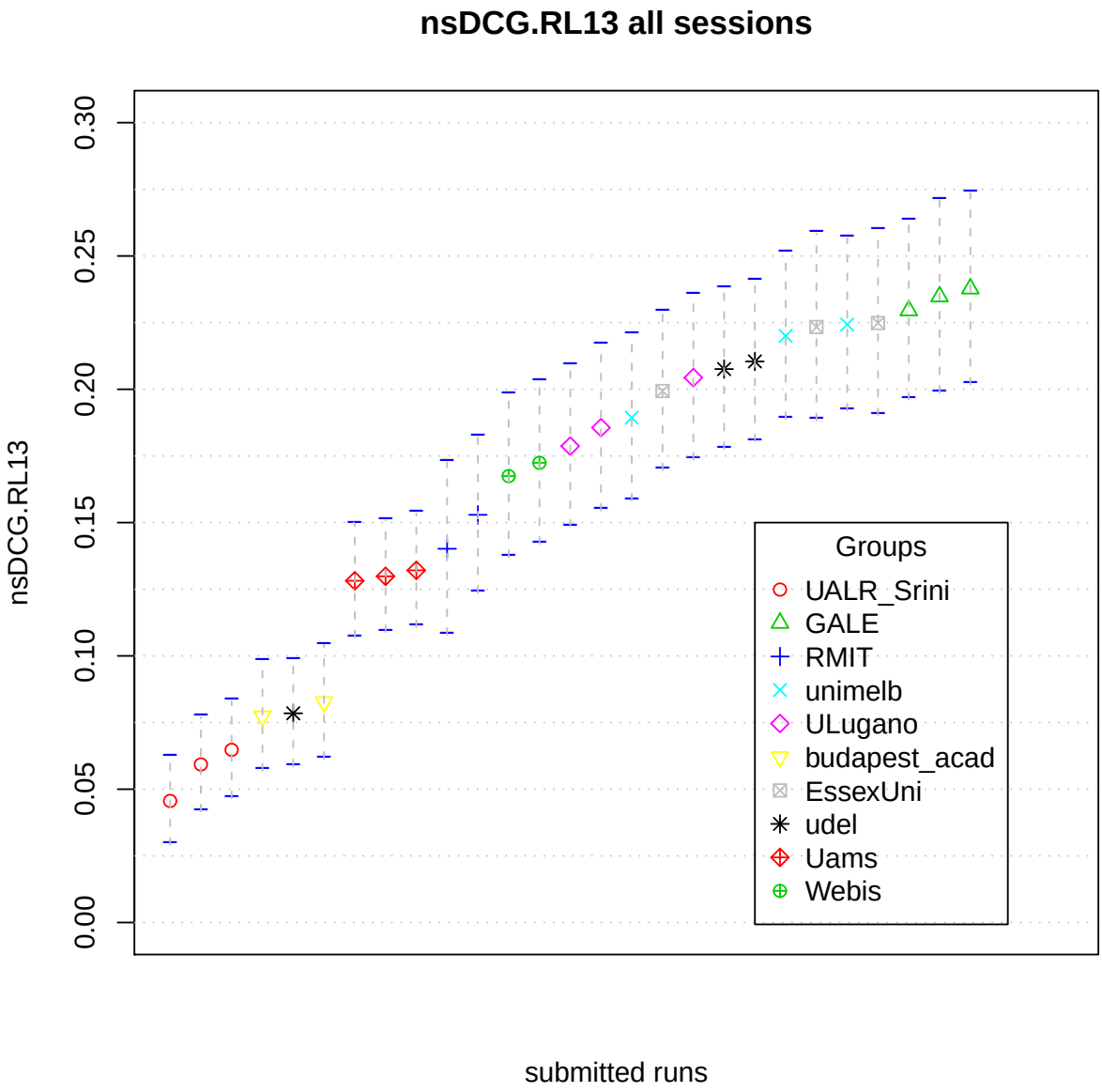

Figure 1: Evaluation scores based on nsDCG@10 for all submitted runs for the ranked lists RL1 → RL3. 95% confidence intervals of the mean nsDCG@10 scores are also depicted in the plot.

Run	snDCG@10.RL13			
	all sessions	Specification	Generalization	Drifting
CengageS10R1	0.2377	0.1827	0.2606	0.2723
CengageS10R2	0.2348	0.1870	0.2528	0.2665
CengageS10R3	0.2295	0.1631	0.2577	0.2708
essex3	0.2249	0.1481	0.2531	**0.2763**
UM10SibmA	0.2243	0.1445	0.2495	**0.2816**
essex1	0.2233	0.1456	**0.2538**	**0.2738**
UM10SibmbB	0.2200	0.1424	0.2434	0.2767
udelIndriASF	0.2104	**0.1605**	0.2362	0.2373
udelIndriB	0.2076	**0.1570**	0.2190	**0.2480**
USIRR2010	0.2044	**0.1532**	**0.2648**	0.2015
essex2	0.1993	0.1395	0.2190	**0.2416**
UM10SimpA	0.1894	0.1294	0.2221	0.2202
USIML052010	0.1856	0.1200	**0.2499**	0.1937
USIML092010	0.1787	0.1149	**0.2419**	0.1860
webis2010w	0.1724	0.1002	0.1890	**0.2299**
webis2010	0.1674	0.1114	0.1763	**0.2156**
RMITBase	0.1529	0.0939	0.1701	**0.1966**
RMITExp	0.1402	0.1199	0.1493	0.1523
uvaExt1	0.1321	0.1035	0.1694	0.1273
uvaExt2	0.1299	0.1108	0.1569	0.1247
uvaExt3	0.1282	0.1029	0.1666	0.1190
bpacad10s1	0.0827	0.0761	0.0819	0.0900
udelIndriA	0.0785	0.0612	0.0797	0.0946
bpacad10s2	0.0774	0.0653	0.0896	0.0785
CARDBNG	0.0648	0.0599	0.0553	0.0781
CARDWIKI	0.0593	0.0388	0.0552	0.0834
CARDWNET	0.0456	0.0208	0.0426	0.0731

Table 2: The snDCG@10 scores for the session RL1→RL3 over all sessions and the specification, generalization and drifting sessions in isolation. Runs are sorted by snDCG@10.RL13 for all sessions. Some of the runs whose ranking over particular reformulation types appears to significantly diverge from the over all sessions ranking are marked by boldface.

6.2 Session Evaluation: G1

In this section we present the results towards the goal (**G1**) of the session track. That is, whether retrieval systems can utilize the history of user requests to increase their performance regarding the current request. The history of user requests consists only of a single query, the initial query, while the current request is the query reformulation. For each topic participants have submitted a ranked list over the query reformulation ignoring the initial query (RL2) and a ranked list over the the query reformulation when the initial query is taken into account. Comparing the performance of the retrieval systems between RL2 and RL3 can show us whether retrieval systems could perform better when they utilized the initial query. Table 3 shows the snDCG@10 and snDCG_dupes@10 scores for RL1→RL2 and RL1→RL3 sessions along with the nDCG@10 scores for RL1, RL2 and RL3. The ↑ symbol denotes an increase in the performance of the retrieval system when utilizing the initial query compared with the performance of the system when ignoring the initial query, the ⇑ symbol denotes a statistically significant increase (when t-test applied), the ↓ symbol denotes a drop in the performance while the ⇓ symbol denotes a significant drop. Systems are ordered by the nsDCG@10.RL12. The same results can be viewed in Figures 5, 7, and 8. The ordering of the systems in the three figures is by nsDCG@10.RL12, nsDCG_dupes@10.RL12 and nDCG@10.RL2, respectively.

Focusing on the nsDCG@10 RL12→RL13 column 11 out of 27 runs improved their performance when the initial query was considered compared to when it was ignored. Out of these 11 systems, only the RMITBase system improved its performance significantly (with significance measures by a paired t-test). The essex1 and essex3 submissions also demonstrated an important increase in their performance without however being statistically significant. On the other hand, 8 out of the 16 drops in performance were statistically significant.

Given that some participants chose to remove duplicate documents from RL3 with respect to RL1, we've also computed nsDCG_dupes@10, which considers documents that have previously appeared in RL1[1..10] as non-relevant. The change in the performance of the systems wrt. to this metric can be viewed in the second column of Table 3. An interesting thing to notice is that when comparing the absolute scores of nsDCG@10 and nsDCG_dupes@10 it appears that considering duplicate documents as non-relevant actually increases the performance of the retrieval systems. This is an artifact which comes from the fact that nsDCG@10 and nsDCG_dupes@10 use different normalization (sDCG of the ideal ranked list). In principle the ideal sDCG@10 is always larger than the ideal sDCG_dupes@10, as we have described in section 5.2. Looking at the sDCG@10 scores instead one can see that there is a number of duplicate documents retrieved over the lists RL2 and RL3. When comparing the nsDCG@10 and nsDCG_dupes@10 columns of the table there are no disagreements regarding the change in the system performance when utilizing the initial query and when not, apart from one, denoted with an asterisk (∗) at the left-most column of the table. There were at least two submitted runs in which RL3 was simply the results of RL2 excluding the results of RL1 (UM10SibmbB and essex2). As it can be viewed in the table penalizing duplicate documents did not alter the change in the performance of these systems between RL2 and RL3. We hypothesis that this is due to the fact that both submissions removed duplicate documents wrt to RL1[1..2000] instead of RL[1..10] making it extremely hard to find so many novel relevant documents in the collection. (For more details one should look at the individual reports of the University of Melbourne and the University of Essex.)

There are more disagreements between the nsDCG@10 RL12→RL13 and the nDCG@10 RL2→RL3. Opposite to the disagreements between the nsDCG@10 and the nsDCG_dupes@10, these disagreements are due to an "anomaly" of the snDCG metric, already described in section 5.2. As already mentioned, the instantiation of nsDCG used here discounts RL2[1..10] and RL3[1..10] as if they were at rank 11 to 20 of RL1 (with an extra discounting due to the reformulation). The nDCG@10 on the other hand discounts RL2 and RL3 as usual, i.e. rank=1..20. Thus, for instance if in RL2 the system has retrieved slightly less relevant documents than in RL3 but slightly higher in the ranked list then the nDCG may penalize RL3 more than snDCG does and so when looking at nDCG one may see a drop from RL2→RL3, while you see an increase from RL2→RL3 when looking at nsDCG. To demonstrate this phenomenon assume the following two hypothetical RL2 and RL3 rankings.

Submitted	nsDCG@10			nsDCG_dupes@10			nDCG@10				
Run	RL12	→	RL13	RL12	→	RL13	RL1	RL2	→	RL3	
UM10SibmA	0.2506	⇓	0.2243	0.2605	⇓	0.2326	0.2391	0.2631	⇓	0.1811	
UM10SibmbB	0.2506	⇓	0.2200	0.2605	⇓	0.2283	0.2391	0.2631	⇓	0.1698	
CengageS10R1	0.2355	↑	0.2377	0.2451	↑	0.2478	0.2177	0.2616	↓	0.2604	†
CengageS10R2	0.2329	↑	0.2348	0.2424	↑	0.2448	0.2147	0.2599	↓	0.2572	†
CengageS10R3	0.2291	↑	0.2295	0.2385	↑	0.2390	0.2096	0.2576	↑	0.2579	
essex3	0.2154	↑	0.2249	0.2230	↑	0.2327	0.2077	0.2215	↑	0.2461	
essex1	0.2154	↑	0.2234	0.2230	↑	0.2309	0.2077	0.2215	↑	0.2353	
essex2	0.2154	⇓	0.1993	0.2230	⇓	0.2060	0.2077	0.2215	⇓	0.1700	
UM10SimpA	0.2143	⇓	0.1894	0.2237	⇓	0.1980	0.2019	0.2341	⇓	0.1625	
udelIndriASF	0.2098	↑	0.2104	0.2200	↑	0.2201	0.1956	0.2288	↓	0.2282	†
udelIndriB	0.2065	↑	0.2076	0.2160	↑	0.2170	0.1898	0.2308	↑	0.2358	
USIRR2010	0.2043	↑	0.2044	0.2150	↓	0.2149	0.1895	0.2145	↑	0.2147	∗
USIML052010	0.2043	⇓	0.1856	0.2150	⇓	0.1948	0.1895	0.2145	⇓	0.1649	
USIML092010	0.2043	⇓	0.1787	0.2150	⇓	0.1875	0.1895	0.2145	⇓	0.1455	
webis2010w	0.1796	↓	0.1724	0.1872	↓	0.1797	0.1638	0.2014	↓	0.1776	
webis2010	0.1796	⇓	0.1674	0.1872	⇓	0.1747	0.1638	0.2014	⇓	0.1621	
RMITExp	0.1435	↓	0.1402	0.1493	↓	0.1450	0.1289	0.1706	↓	0.1525	
RMITBase	0.1375	⇑	0.1529	0.1427	⇑	0.1584	0.1246	0.1534	↑	0.1834	‡
uvaExt1	0.1358	↓	0.1321	0.1437	↓	0.1399	0.1071	0.1890	↓	0.1773	
uvaExt3	0.1262	↑	0.1282	0.1335	↑	0.1358	0.1071	0.1543	↑	0.1642	
uvaExt2	0.1260	↑	0.1299	0.1334	↑	0.1374	0.1071	0.1623	↑	0.1714	
bpacad10s1	0.0830	↓	0.0827	0.0874	↓	0.0870	0.0632	0.1122	↓	0.1077	
bpacad10s2	0.0830	↓	0.0774	0.0874	↓	0.0812	0.0632	0.1122	↓	0.0997	
udelIndriA	0.0761	↑	0.0785	0.0789	↑	0.0813	0.0626	0.1034	↑	0.1133	
CARDBNG	0.0664	↓	0.0648	0.0690	↓	0.0679	0.0554	0.0900	↓	0.0825	
CARDWIKI	0.0664	↓	0.0593	0.0690	↓	0.0619	0.0554	0.0900	↓	0.0632	
CARDWNET	0.0664	⇓	0.0456	0.0690	⇓	0.0470	0.0554	0.0900	⇓	0.0263	

Table 3: The snDCG@10 and snDCG_dupes@10 scores for RL1→RL2 and RL1→RL3 sessions along with the nDCG@10 scores for RL1, RL2 and RL3. The ↑ symbol denotes an increase in the performance of the retrieval system when utilizing the initial query compared with the performance of the system when ignoring the initial query, the *Uparrow*symbol denotes a statistically significant increase (when t-test applied), the ↓ symbol denotes a drop in the performacen while the ⇓ symbol denotes a significant drop. The † symbol at the left column denotes a disagreement between the nsDCG@10.RL12–to–nsDCG@10.RL13 and the nDCG@10.RL2–to–nDCG@10.RL3 change of performance, a ‡ denotes a disagreement in the significance of the change between the same two metrics while an ∗ symbol denotes a disagreement between the nsDCG@10.RL12–to–nsDCG@10.RL13 and the nsDCG_dupes@10.RL12–to–nsDCG_dupes@10.RL13.

```
RL2 :  R R N N N N N N N N
RL3 :  N N R R R N N N N N
```

The DCG@10.RL2 is 1.4048 while the DCG@10.RL3 is 1.1349, pointing out that RL2 is a better ranking than RL3. The sDCG@10.RL2 (a component of the snDCG@10.RL12) is 0.473, while the sDCG@10.RL3 is 0.662, pointing out that RL3 is actually a better ranking than RL2.

A dagger (†) at the left-most column of the table denotes a disagreement between the nsDCG@10.RL12–to–nsDCG@10.RL13 and the nDCG@10.RL2–to–nDCG@10.RL3 change of performance, while a double dagger (‡) denotes a disagreement in the significance of the change between the same two metrics.

7 Analysis

7.1 System performance and type of reformulation

In this section we do some further analysis of the performance of the retrieval systems with respect to the different types of reformulations.

Average number of relevant documents

The Table 4 shows the number of relevant and highly relevant retrieved on average in RL1[1..10], RL2[1..10] and RL3[1..10], along with the number of relevant and highly relevant novel documents retrieved on average in RL2[1..10] and RL3[1..10] over all sessions, specification sessions, generalization sessions and drifting sessions. As it can be observed the submitted runs retrieved much fewer relevant documents over the initial query for the specialization session than the generalization or the drifting ones. Further there is a clear increase in the number of relevant and highly relevant documents from RL1 to RL2 or RL3, except for the case of drifting sessions. The participating runs appear to retrieve a number of duplicate documents especially in the case RL3, nevertheless most of the retrieved documents in RL2 and RL3 seem to be novel (or at least not exact duplicates).

	RL1		RL2		RL2 Novel		RL3		RL3 Novel	
	rel	hrel	rel	hrel	rel	hrel	rel	hrel	rel	hrel
All sessions	1.265	0.921	1.698	1.102	1.517	0.972	1.569	0.979	1.397	0.872
Specification	0.719	0.529	1.761	1.045	1.485	0.879	1.581	0.888	1.372	0.755
Generalization	1.755	1.012	2.050	1.341	1.774	1.106	1.858	1.250	1.533	1.052
Drifting	1.373	1.231	1.321	0.946	1.321	0.946	1.301	0.827	1.301	0.827

Table 4: Number of relevant and highly relevant retrieved on average in RL1[1..10], RL2[1..10] and RL3[1..10], along with the number of relevant and highly relevant novel documents retrieved on average in RL2[1..10] and RL3[1..10] over all sessions, specification sessions, genralization sessions and drifting sessions.

Session evaluation per reformulation type

The Table 5 reports the snDCG@10 scores for RL1→RL2 and RL1→RL3 sessions. The ↑ symbol denotes an increase in the performance of the retrieval system when utilizing the initial query compared with the performance of the system when ignoring the initial query, the ⇑symbol denotes a statistically significant increase (when t-test applied), the ↓ symbol denotes a drop in the performance while the ⇓ symbol denotes a significant drop.

One can observe a number of patterns across all the runs by examining the improvements and drops of the performance over the three reformulation types. Some of those patterns have been noted by the symbols

at the right-most column of the table. The dagger (†) denotes runs that improved over the specification and generalization sessions but didn't over the drifting, the plus (+) denotes runs that improved over the generalization and drifting but not over the specification, the bullet (•) denotes runs that improved over generalizations only, the asterisk (∗) denotes runs that improved over the specification only, while the tick (√) denotes runs that improved over all types.

Figures 5–8 at the end of this document also illustrate the change of the system performance as measures by nsDCG@10, nsDCG_dupes@10 and nDCG@10.

Submitted Runs	nsDCG@10												
	all sessions			Specification			Generalization			Drifting			
	RL12	→	RL13	RL12	→	RL13	RL12	→	RL13	RL12	→	RL13	
UM10SibmA	0.2506	⇓	0.2243	0.1674	⇓	0.1445	0.2804	⇓	0.2495	0.3073	⇓	0.2816	
UM10SibmbB	0.2506	⇓	0.2200	0.1674	⇓	0.1424	0.2804	⇓	0.2434	0.3073	⇓	0.2767	
CengageS10R1	0.2355	↑	0.2377	0.1756	↑	0.1827	0.2658	↓	0.2606	0.2683	↑	0.2723	†
CengageS10R2	0.2329	↑	0.2348	0.1793	↑	0.1870	0.2590	↓	0.2528	0.2632	↑	0.2665	†
CengageS10R3	0.2291	↑	0.2295	0.1619	↑	0.1631	0.2588	↓	0.2577	0.2697	↑	0.2708	†
essex3	0.2154	↑	0.2249	0.1563	↓	0.1481	0.2381	↑	0.2531	0.2542	↑	0.2763	+
essex1	0.2154	↑	0.2233	0.1563	↓	0.1456	0.2381	↑	0.2538	0.2542	↑	0.2738	+
essex2	0.2154	⇓	0.1993	0.1563	⇓	0.1395	0.2381	↓	0.2190	0.2542	⇓	0.2416	
UM10SimpA	0.2143	⇓	0.1894	0.1546	⇓	0.1294	0.2452	⇓	0.2221	0.2463	⇓	0.2202	
udelIndriASF	0.2098	↑	0.2104	0.1570	↑	0.1605	0.2376	↓	0.2363	0.2377	↓	0.2373	∗
udelIndriB	0.2065	↑	0.2076	0.1552	↑	0.1567	0.2206	↓	0.2190	0.2451	↑	0.2480	†
USIRR2010	0.2043	↑	0.2044	0.1529	↑	0.1532	0.2702	↓	0.2648	0.1969	↑	0.2015	†
USIML052010	0.2043	⇓	0.1856	0.1529	⇓	0.1200	0.2702	⇓	0.2499	0.1969	↓	0.1937	
USIML092010	0.2043	⇓	0.1787	0.1529	⇓	0.1149	0.2702	⇓	0.2419	0.1969	↓	0.1860	
webis2010w	0.1796	↓	0.1724	0.1097	↓	0.1002	0.1868	↑	0.1889	0.2430	↓	0.2299	•
webis2010	0.1796	⇓	0.1674	0.1097	↑	0.1114	0.1868	↓	0.1763	0.2430	⇓	0.2156	
RMITExp	0.1435	↓	0.1402	0.1334	↓	0.1199	0.1438	↑	0.1493	0.1533	↓	0.1523	•
RMITBase	0.1375	⇑	0.1529	0.0932	↑	0.0939	0.1501	⇑	0.1701	0.1705	↑	0.1966	√
uvaExt1	0.1358	↓	0.1321	0.1153	↓	0.1035	0.1689	↑	0.1694	0.1267	↑	0.1273	
uvaExt3	0.1262	↑	0.1282	0.1044	↓	0.1029	0.1492	↑	0.1666	0.1273	↓	0.1190	
uvaExt2	0.1260	↑	0.1299	0.1040	↑	0.1109	0.1558	↑	0.1569	0.1213	↑	0.1247	√
bpacad10s2	0.0830	↓	0.0774	0.0760	↓	0.0653	0.0821	↑	0.0896	0.0908	↓	0.0785	•
bpacad10s1	0.0830	↓	0.0827	0.0760	↑	0.0761	0.0821	↓	0.0819	0.0908	↓	0.0900	
udelIndriA	0.0761	↑	0.0785	0.0592	↑	0.0612	0.0811	↓	0.0797	0.0884	↑	0.0946	†
CARDBNG	0.0664	↓	0.0648	0.0492	↑	0.0599	0.0656	↓	0.0553	0.0841	↓	0.0781	
CARDWIKI	0.0664	↓	0.0593	0.0492	↓	0.0388	0.0656	↓	0.0552	0.0841	↓	0.0834	
CARDWNET	0.0664	↓	0.0456	0.0492	⇓	0.0208	0.0656	⇓	0.0426	0.0841	↓	0.0731	

Table 5: The snDCG@10 scores for RL1→RL2 and RL1→RL3 sessions. The ↑ symbol denotes an increase in the performance of the retrieval system when utilizing the initial query compared with the performance of the system when ignoring the initial query, the $Uparrow$symbol denotes a statistically significant increase (when t-test applied), the ↓ symbol denotes a drop in the performacen while the ⇓ symbol denotes a significant drop. The † symbol denotes runs that improved over the specification and generalization sessions but didn't over the drifting, the + symbol denores runs that improved over the generalization and drifting but not over the specification, the • symbol denotes runs that improved over generalizations only, the ∗ symbol denotes runs that improved over the specification only, while the √ symbol denotes runs that improved over all types.

*** ANOVA

7.2 Analysis over queries

Figures 2 illustrate the initial query hardness as measured by the average nDCG@10.RL1 over all queries. Figure 3 illustrates the query hardness both of the initial query (left-hand side plot) and the query reformu-

lation (right-hand side plot) as measured by the average nDCG@10.RL1 and ndcg@10.RL2 over the three reformulation types. As it can be observed by the two figures the initial queries were in general hard queries. The initial queries of the specialization sessions appear to be the hardest ones, and thus they offer a larger margin of improvement of performance over the reformulated query. On the contrary the initial query of generalization sessions seems to be well specified, even though an improvement over the reformulated query can be observed on average.

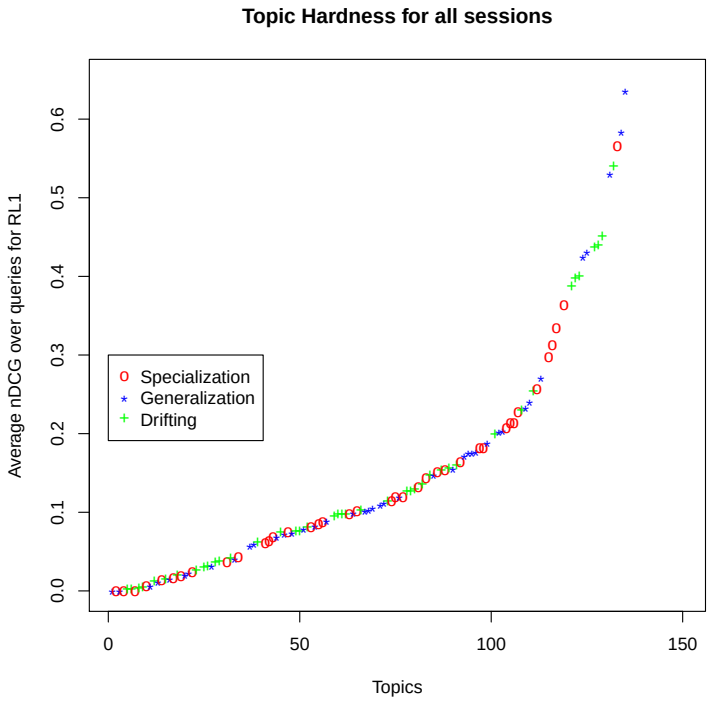

Figure 2: Query hardness measured by the average nDCG@10 over all session.

Discounting and logarithm bases

In the framework used to define typical nDCG metric, relevance scores are mapped to relevance grades, e.g. a score of 3 is given to highly relevant documents, a score of 2 to fairly relevant documents and so on. Relevance scores are viewed as the *gain* returned to a user when examining the document. Thus, the relative value of relevance scores dictates how more valuable for instance a highly relevant document is to the user than a marginally relevant. Even though, relevance scores were used directly as gains when nDCG was originally defined, alternative gain functions that map gain values to relevance scores have been used. To account for late arrival of relevant documents, gains are then discounted by a function of the rank. The discount function is viewed as an indication of the patience of a user to step down the ranked list of documents. In other words, it expresses the probability of a user seeing a document at a certain rank. The discounted gains are then summed progressively from rank 1 to k and this discounted cumulative gain is normalized to the range of 0 to 1, resulting in the *normalized discounted cumulative gain* (nDCG).

The original instantiation of nDCG, as it appeared in Järvelin and Kekäläinen [**?**], can be computed as,

$$nDCG@k = \frac{DCG@k}{optDCG@k} \quad \text{where} \quad DCG@k = \sum_{i=1}^{b} rel(1) + \sum_{i=b+1}^{k} rel(i)/\log_b(i)$$

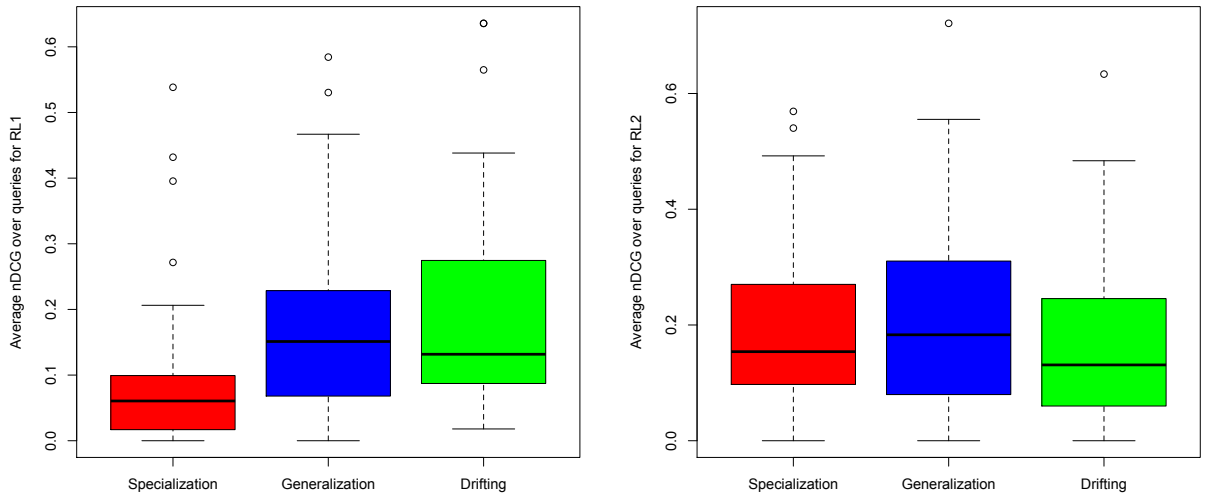

Figure 3: Query hardness measured by the average nDCG@10 over initial query (RL1) and query reformulation (RL2) for the three reformulation types.

where $rel(i)$ is the relevance score of the document at rank i, k is an arbitrary cut-off rank and optDCG denotes the DCG value for an ideal ranked list. The logarithm base b controls the severity of the discount function; the smaller the base the less patient the user modeled is. The base 2 logarithm is the one that has mostly appeared in the literature.

Note that the afore-described instantiation of nDCG does not discount the first $b+1$ documents in the ranked list. To cope with this a number of different discount functions have been used. Similar to discount functions a number of different gain functions have also appeared in the literature. For instance, in Burges et al. [**?**] DCG was computed as,

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel(i)-1}}{log_2(i+1)}$$

while in Järvelin et al. [**?**] DCG was computed as,

$$DCG@k = \sum_{i=1}^{k} \frac{rel(i)}{(1+log_2(i))}$$

The session-based nDCG has been proposed to incorporate multiple queries in an interactive retrieval scenario, in which a user moves down a ranked list of documents and at some cut-off rank, she/he reformulates the query. In that case DCG is computed at the reformulation cut-off for each query and the stopping cut-off for the last reformulation. In the case of Sessions Track, the reformulation cut-off is not known since it is not an interactive track. Given that a fixed reformulation cut-off has been selected at rank 10. That is, a user is assumed to reformulate his query after observing the document at rank 10.

A question that needs to be answered is how should one select the logarithm bases. As mentioned earlier, in the case of nDCG the logarithm base captures different search scenarios (e.g. precision vs. recall-orient retrieval, etc.) and users (patient vs. impatient users). In the case of nsDCG the logarithm bases play

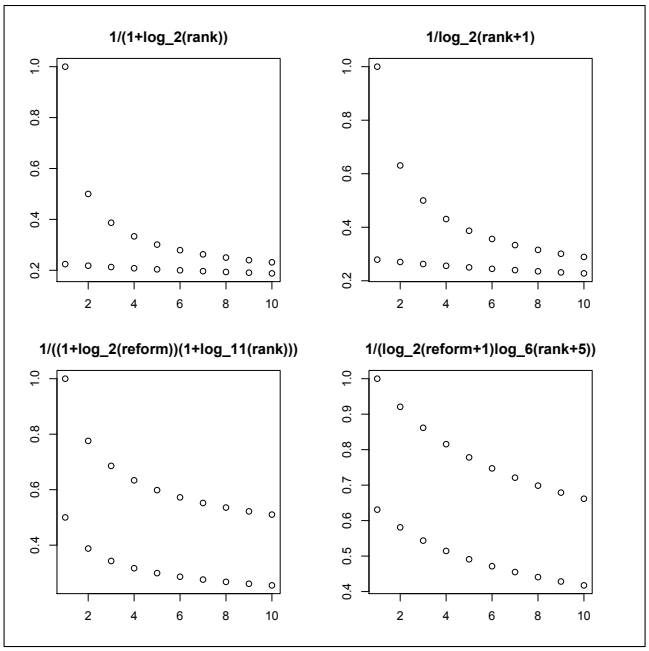

Figure 4: Different discounts for top 10 documents in the first query and the query reformulation. The discounts are shown for the two queries in isolation, starting from rank 1 in both cases.

a similar role, nevertheless there is a constraint of how to set the bases. Under a realistic scenario, the documents in the ranked list for query q should not receive discounting larger the documents in the ranked list for query $q + m$, for $m \geq 1$.

Given that the reformulation cut-off in Sessions Track is set at rank 10, the maximum penalty received by a document in query q is the penalty of the last document before the reformulation, i.e. the document at rank 10. The penalty that this document receives is,

$$(1 + log_b 10) * (1 + log_{bq} q)$$

The minimum penalty that a document receives for a query $q + m$ is the penalty that the top document of the query $q + 1$ receives, thus the penalty of that document is,

$$(1 + log_b 1) * (1 + log_{bq}(q + 1))$$

Since we only have two queries in the session track, $q = 1$ and thus $q + 1 = 2$. By replacing the q's in the above formulas we get that

$$(1 + log_b 10) * (1 + log_{bq} 1) \quad \text{should be less than} \quad (1 + log_b 1) * (1 + log_{bq} 2)$$

Solving the inequality for b and setting $bq = 2$ results in $b > 10$. By repeating the above calculations for the discount function in Burges et al. [**?**] we find that for $bq = 2$, b should be at least 6.

As one can notice the afore-derived logarithm bases are much larger than the typical base of 2 used in most implementations of nDCG. An alternative instantiation of a session-based nDCG could be defined as follows: since we assume that a user will never look beyond rank 10 in any query we can concatenate the top 10 documents of the query $q + 1$ at the bottom of the top 10 documents of the query q and essentially simply compute nDCG@(10*q). In the case of the session track this is nDCG@20. Naturally, documents after the

user's reformulation will be penalized more than the ones before the reformulation and we can select the base of the discounting logarithm without any constraints. Moreover we can further penalize the reformulated ranked list by the position of the query in a series of reformulations as in [**?**].

Nevertheless, in this instantiation of the session-based nDCG the relative discount between two consecutive documents is different for each query. This can be view in Figure 7.2. The discounts of the RL1[1..10] and RL2/RL3[1..10] are shown in isolation to make the comparison of the relative drop of the weight for each document in the two ranked lists.

8 Conclusions

Some preliminary coclusions:

*** Most retrieval systems could not utilize the initial query to improve their performance over the reformulated query.

*** There is a strong "system–reformulation type" effect.

*** Generalization sessions seem to have been well specified from the initial query instead of over-specified; this leads to the conclusion that we need to re-examine the manner reformulations are constructed. Nevertheless, given the larger number of relevant documents found in the generalization sessions by all systems and the small number of relevant documents found per each system, it is obvious that there is a large margin of potential improvements for all sessions.

A Descriptions of Submitted Runs

Each of the methods used by each one of the groups that participated in the track is summarized below. For further details on the techniques used refer to the individual groups reports for the Session Track.

Bauhaus University Weimar: The Webis group from the Bauhaus University Weimar submitted two runs, `webis2010` and `webis2010w`. The participants took a two steps approach. In a preprocessing phase they used query segmentation, comparing possible query segments against the Google n-gram collection. In the second phase queries were submitted to the Carnegie Mellon ClueWeb search engine and run against the Category B ClueWeb collection. The `webis2010` run applied a maximum keyword framework in which for each query or query session all query terms are considered and the longest query that has a reasonable number of hits (between 1 and 1000) is selected to better represent the user's information need. In the case that all queries returned more than 1000 results then the complete query containing all terms along a session was used. The `webis2010w` run used different term weights in the Indri query language to cope with case where query terms were added or deleted during the query reformulation. Query terms that only appeared in the first query were given a weight of 0.5, those that appeared only in the second query were given a weight of 2 and those that appeared in both queries were given a weight of 1. It was noticed that due to short sessions the maximum query algorithm would often times select all query terms as the maximum query.

Gale, Cengage Learning: The Cengage Learning group submitted three runs, `CengageS10R1`, `CengageS10R2`, and `CengageS10R3`. The participants indexed the Category B subset of the ClueWeb09 collection using Lucene. A number of different techniques were then used for their submissions:

Query Term Weighting : Query term weights we applied depending upon which query the occurred in. Query terms were divided into three different categories: terms that appear only in the first query, terms that appear only in the second query, and terms that appear in both queries. Through experimentation, it was found that the best weights for these three groups were dependent upon the type of reformulation. This approach necessitated the ability to automatically categorize query pairs. A number of different techniques were used for this purpose.

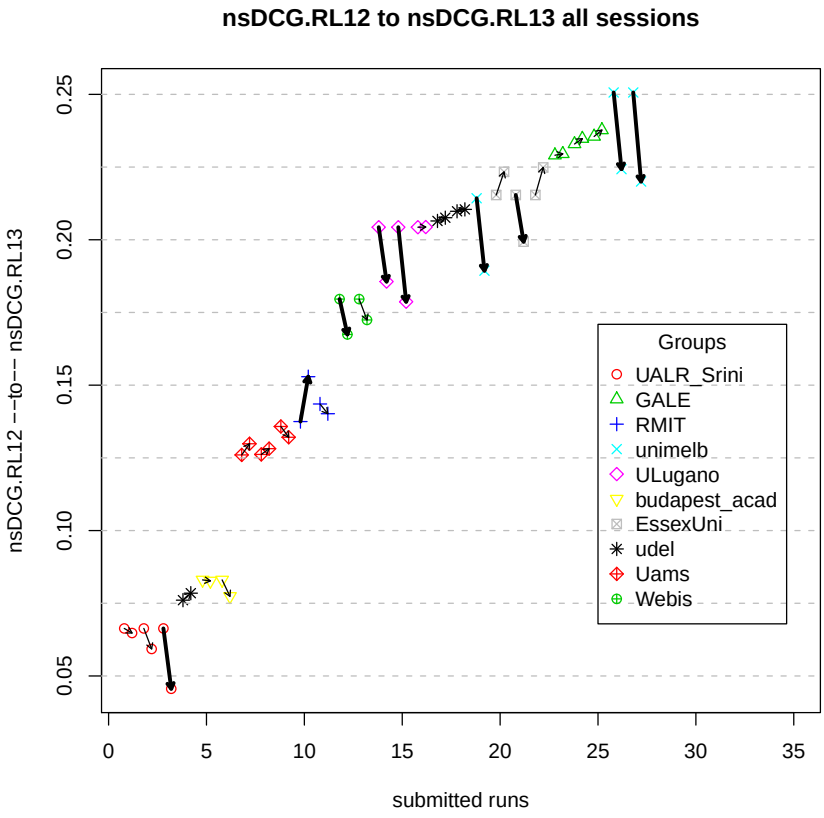

Figure 5: Evaluation scores based on nsDCG@10 for all submitted runs for the ranked lists RL1 → RL2 and RL1 → RL3. Arrows indicate the change in the evaluations score between the two ranked lists for each one of the submitted runs. Thick solid arrows indicate statistically significant changes according to paired two-sided Student t-test.

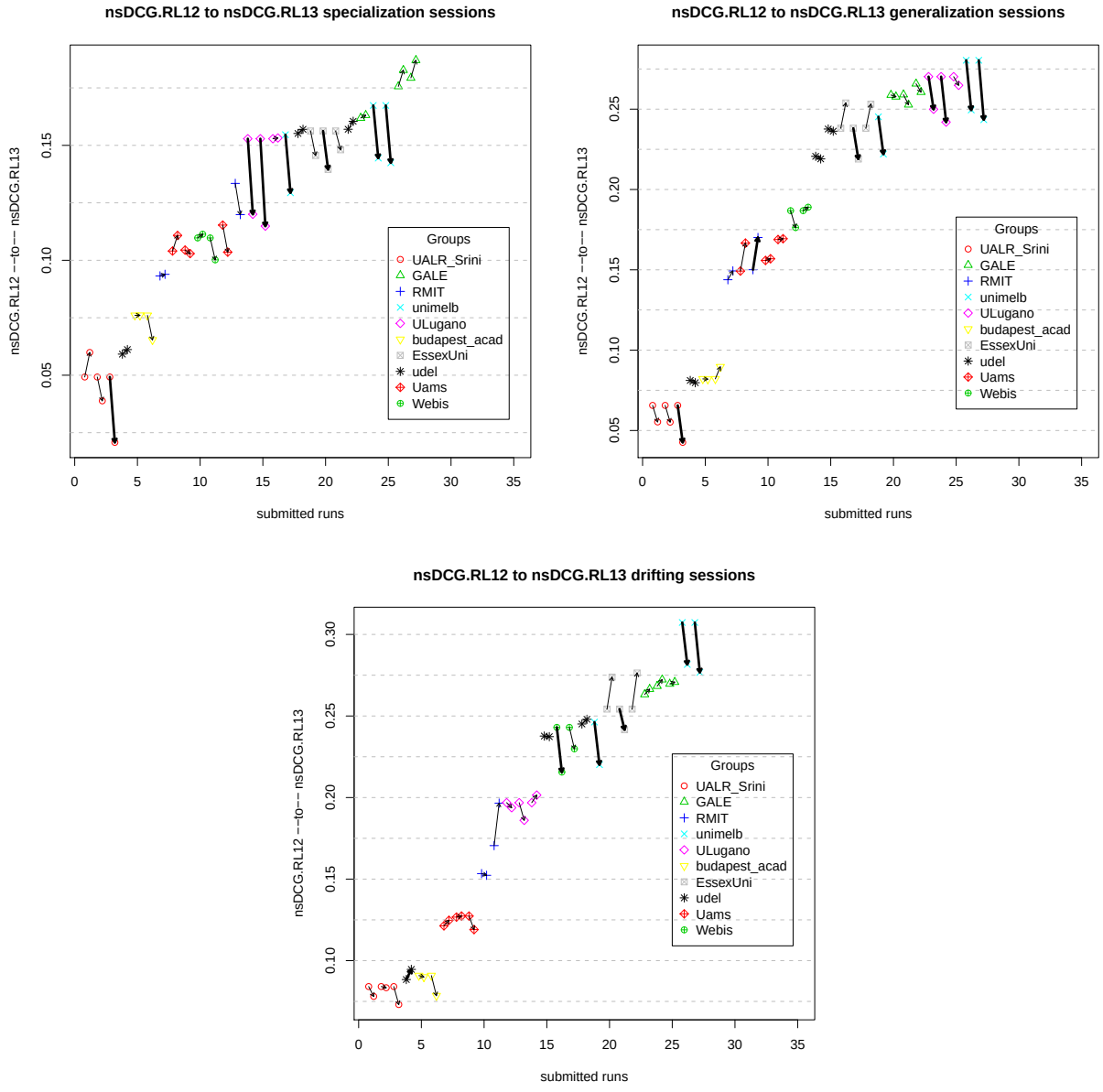

Figure 6: Evaluation scores based on nsDCG@10 for all submitted runs for the ranked lists RL1 → RL2 and RL1 → RL3 for specification, generalization and drifting-reformulation sessions.

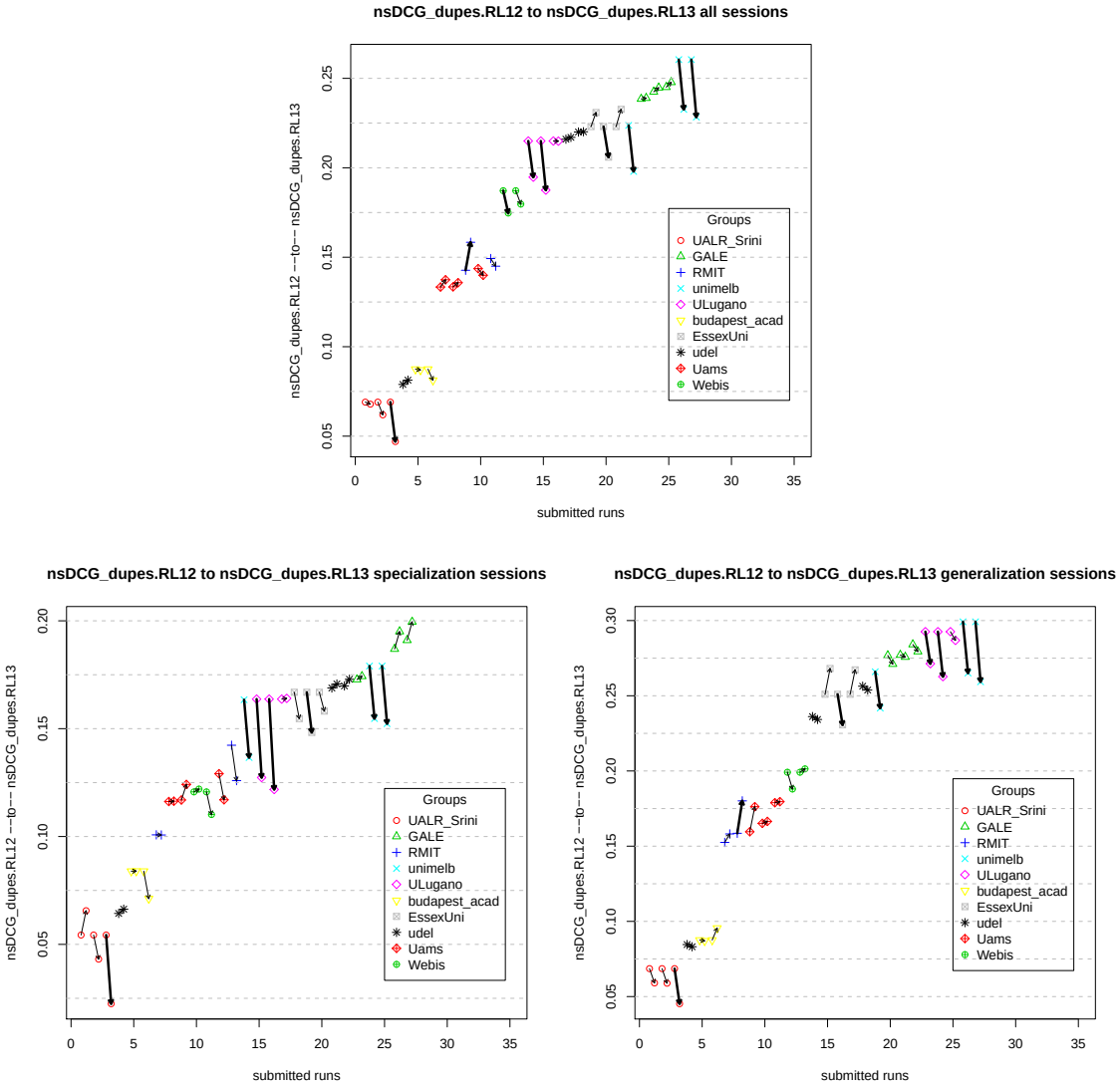

Figure 7: Evaluation scores based on nsDCG_dupes@10 for all submitted runs for the ranked lists RL1 → RL2 and RL1 → RL3. Duplicate documents wrt. RL1 are considered non-relevant. Arrows indicate the change in the evaluations score between the two ranked lists for each one of the submitted runs. Thick solid arrows indicate statistically significant changes according to paired two-sided Student t-test. Scores are also shown for specification and generalization-reformulation sessions. The ones for the drifting-reformulation sessions are exactly the same as the nsDCG@10.

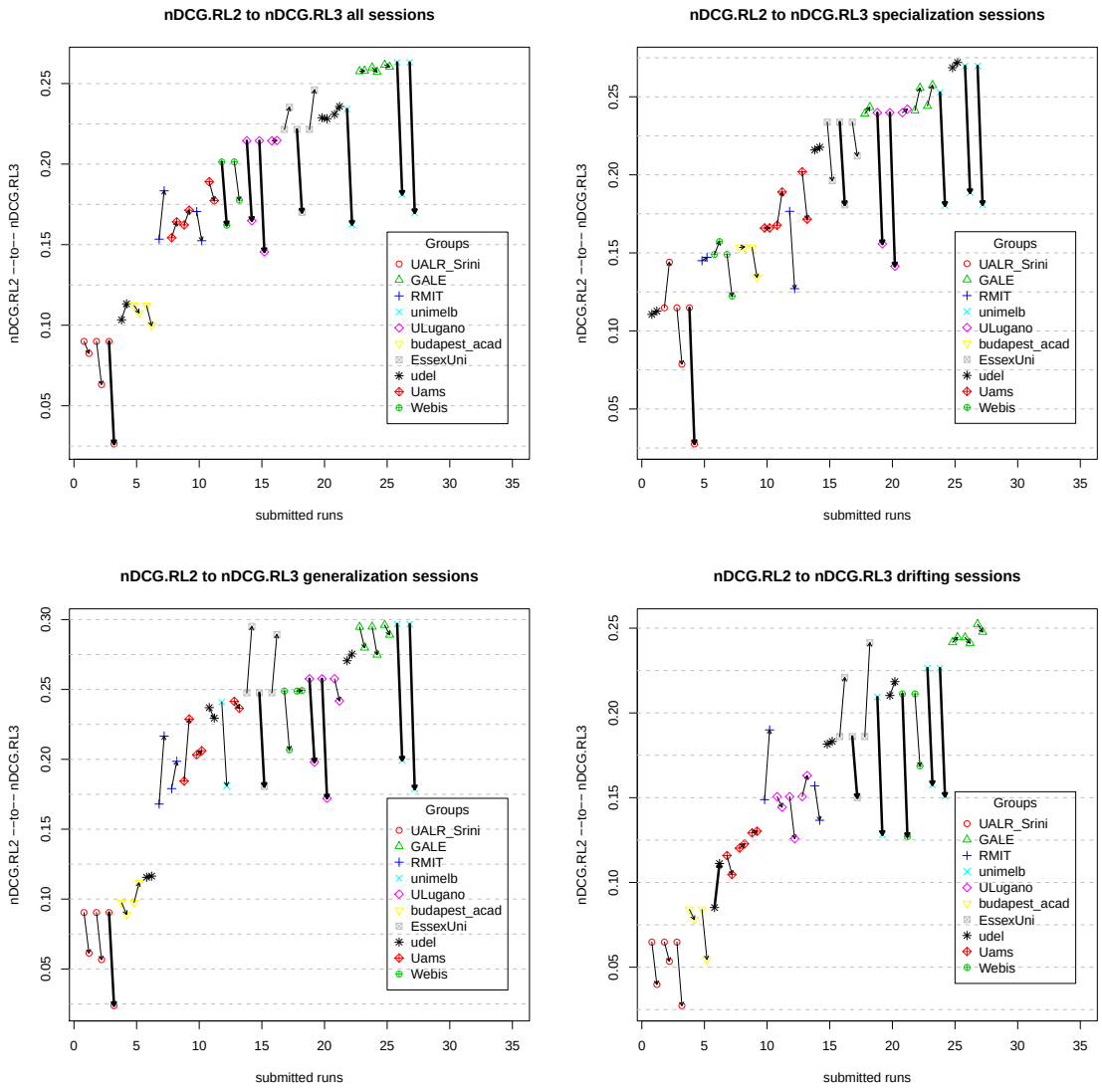

Figure 8: Evaluation scores based on nDCG@10 for all submitted runs for the ranked lists RL2 and RL3. Arrows indicate the change in the evaluations score between the two ranked lists for each one of the submitted runs. Thick solid arrows indicate statistically significant changes according to paired two-sided Student t-test. Scores are also shown for specification, generalization and drifting-reformulation sessions.

Category Re-ranking : Query and documents were first classified over the Open Directory Project (ODP) categories. Every category in the ODP was indexed against its title and the descriptions of all the pages categorized under it. To categorize a query or document, the text of that query or document was submitted to the ODP index and a list of search results was returned with a retrieval score for each result. The top ten results were selected as the best category matches for the query and were given a weight proportional to the retrieval score returned by Lucene.

Query Expansion : (a) Usage Log Query Expansion : For each query a list of related search terms was produced by mining the usage logs from Cengage Learning products. The term collocation formula was similar in concept to the tf-idf weighting scheme, in that it rewards frequently co-occurring terms, but minimizes the impact of the most common search terms. Expansion terms were sought for all possible sub-queries, with expansion terms for longer phrases receiving a bonus based on that length. (b) Corpus Collocation Query Expansion : Corpus-based collocation expansion was done very similarly to usage log-based collocation expansion. The major difference is in how the expansion terms were collected. Cengage Learning maintains a collocation database. This database was compiled against large portions of Cengage Learning's digital material and could return a list of the fifty most common words and phrases that appear near a given term.(c) WordNet Expansion : The words in the query needed to be resolved as to which sense of the word they referred by determining which sense of the word is most similar to the other words in the query. We created a measure which incorporated WordNet's tag count, a measure of how often each sense was encountered in the tagging of corpora.

The `CengageS10R1` submission used Query Term Weighting and Corpus terms collocation expansion, the `CengageS10R2` used Term weighting, Usage-log expansion, Corpus collocation expansion, Pseudo-relevance expansion and the `CengageS10R3` used WordNet expansion and Category re-ranking.

Hungarian Academy of Science: The Hungarian Academy of Science group submitted two runs, `bpacad10s1` and `bpacad10s2`. The entire ClueWeb corpus was used. For the RL1 and RL2 the AND Boolean operator was used to construct a new query out of the original query and its reformulation and documents were ranked by Okapi BM25. For the RL3, documents in the RL2 list were re-ranked based on the reformulation type and whether they occurred in RL1 or not – `bpacad10s1`. The reformulation type was determined from the surface form of the question. The `bpacad10s2` submission was produced by the weighted union of the RL1 and RL2 result lists.

RMIT University: The RMIT group submitted two runs, `RMITBase` and `RMITExp`. The experiments were run on the ClueWeb Category B collection. Indexing and searching was done with the Lemur toolkit (v 4.12). Dirichlet-smoothed language model was used for ranking. In the `RMITBase` submission for the RL1 and RL2 the first query and its reformulation was used respectively. For the RL3 the union of the query terms from both queries were used. In the `RMITExp` the queries (first query, reformulation and union of their terms) were first submitted to Google to obtain "related search" suggestions. The union of the query terms of the Google query suggestions and the three aforementioned queries were then run against the ClueWeb dataset.

The University of Melbourne: The University of Melbourne group submitted three runs, `UM10SimpA`, `UM10SibmA`, and `UM10SibmB`, using the entire ClueWeb09 corpus. Regarding the retrieval methods for RL1 and RL2, the `UM10SimpA` was a content-only run, and original impact model was employed for similarity computation. For the other two runs, the impact-based version of BM25 was employed. Moreover, the similarity score was a combination of content (50%), incoming anchor (25%) and PageRank (25%) scores. For all three runs, the documents with Waterloo spam scores of 30% or less were discarded from the results. The RL3 result were just the merging of RL1 and RL2, with two merging methods - A and B. In the merging method A, which was applied to `UM10SimpA` and `UM10SibmA`, a similarity degree s between the two respective queries was calculated, and the score S3 is $S2 - s * S1$, where S1, S2, S3 are scores for RL1, RL2 and RL3, respectively. In the merging method B, which applied to `UM10SibmB`, S3 was simply $S2 - S1$.

University of Amsterdam: The University of Amsterdam group submitted three runs, `uvaExt1`, `uvaExt2`, and `uvaExt3` using the entire ClueWeb09 corpus. The experiments by UAms focused on the use of blind relevance feedback to bias a follow-up query towards or against the topics covered in documents that were

returned to the user in response to the original query. Blind relevance feedback takes the most discriminative terms from a set of documents retrieved for a query, and uses these to build a query model that incorporates information about the topic underlying the documents. These experiments followed the intuition that for original queries, when no context for disambiguation is available, diverse result lists should be presented, that have a high chance of answering any aspect of the information need underlying a query. Once more context is available, this is used to bias results towards the relevant aspects of the query using blind relevance feedback. Three methods for biasing results for the follow-up query were explored. First, it was assumed that results returned for the original query were helpful and can be used to focus or disambiguate results for the follow-up query. Thus, feedback terms extracted from the top-ranked documents of the original query were used to expand the follow-up query – `uvaExt1`. Second, the assumption that results for the original query were not helpful was covered. The set of expansion terms generated from the top-ranked documents returned for the follow-up query was taken as a base set, and the feedback terms that were generated using the original query were removed – `uvaExt2`. Finally, UAms considered that the underlying topic may best be represented by both queries, and used the feedback terms generated by both queries to expand the follow-up query – `uvaExt3`. The system was implemented based on the Indri retrieval engine. Retrieval runs for the original queries (RL1) were generated by interpolating language modeling and phrase-based retrieval scores. Based on those runs, diversification was performed using the maximum marginal relevance method (MRR) with clusters obtained by latent Dirichlet allocation (LDA). For the follow-up queries where no additional context is taken into account (RL2), three different methods are explored: (1) language modeling + phrase-based retrieval – `uvaExt1`, (2) diversification using pseudo-relevance feedback – `uvaExt2`, and (3) diversification as for the original query (using MRR and LDA) – `uvaExt3`. From these individual runs, runs that combine information using the original and the follow-up query (RL3) are generated using the pseudo-relevance methods described above.

University of Arkansas at Little Rock: The University of Arkansas group submitted three runs using the full ClueWeb09 corpus, `CARDBNG`, `CARDWIKI`, and `CARDWNET`. No query processing was performed for RL1 and RL2. Regarding RL3, the `CARDBNG` submission used the Bing search engine to categorize the reformulation type (generic/specific/drifting etc.) and used query expansion according to categorization above. The `CARDWIKI` used the Wikipedia search engine to categorize the reformulation type query expansion according to the categorization. The `CARDWNET` used the Wordnet dictionary for query expansion of the intersection and the union of the original query and its reformulation query terms.

University of Delaware: The University of Delaware submitted three runs using the same baseline retrieval method on three different subsets of the ClueWeb09 collection: the full Category A set, the Category A set filtered for spam pages using the Waterloo spam scores, and the Category B set. The RL3 submission estimated a probability that a document in RL2 would be viewed by a user twice (once for the first query, once for the second), and scaled the document's score down accordingly.

University of Essex : The University of Essex submitted three runs, `essex1`, `essex2`, and `essex3`. In all their runs they used the publicly available Carnegie Mellon ClueWeb search engine. For the ranked lists RL1, RL2 queries were submitted as they are to the search engine which in return used the Query Likelihood model to retrieve a ranked list of documents. The Waterloo Spam Rankings for the ClueWeb09 Dataset was used to filter the spam documents from the ranked lists. The `essex1` run is the first baseline method to retrieve ranked list RL3, in which a new query consisting of both queries in the session was submitted to the Indri search engine. The `essex2` run is the second baseline method to retrieve ranked list RL3. It reflects on the assumption that the user is not satisfied with the first set of results and that is why she reformulated her original query. In this baseline the participants use a naive way to utilize the original query by filtering the retrieved documents for the reformulated query in the session. The filtering works simply by eliminating the documents which appear in the first ranked list. In `essex3` a method for extracting useful terms and phrases to expand the reformulated query in the session was used. This method stems from previous work in using query logs to extract related queries and the group's past work in the AutoAdapt project to learn domain models from query logs. Due the lack of availability of query logs an anchor log constructed from the same dataset (the ClueWeb09 category B dataset) was used to simulate the query log. The anchor log was extracted and made publicly available by the University of Twente. Using the anchor log they extract

the top common associated queries of both queries in the session using Fonseca's association rules [**?**]. Then they expanded the reformulated query with the extracted phrases or terms and the original query giving higher weights to the reformulated query.

University of Lugano : The University of Lugano group submitted three runs, `USIML052010`, `USIML092010`, and `USIRR2010`. The Category B subset of the ClueWeb was indexed with the Terrier information retrieval system and used for the retrieval. The ranked lists RL1 and RL2 for all the three submitted runs were generated by using the original query and its reformulation, respectively and scoring documents by the BM25 implementation of Terrier. With respect to RL3, two approaches were used. In the first approach the third ranking (RL3) was generated by scoring documents according to the weighted summation of the reciprocal ranks of documents in RL1 and RL2, where the weight given to documents from RL1 was negative and RL2 was positive. Thus the score for a document d was computed as follows: $score(d, RL3) = \alpha * (1/rank(d, RL1)) + (1 + \alpha) * (1/rank(d, RL2))$. If a document was not present in one of the ranked lists, its reciprocal rank was set to 0. The parameter α was empirically set to 0.2 – `USIRR2010`. In the second approach the relevance model for the first and second query was build using the top N ranked documents from RL1 and RL2 as the pseudo-relevant documents (denoted PR1 and PR2). The relevance models R1 and R2 were estimated by averaging the relative frequencies for terms across the pseudo-relevant documents. The R1 estimates were smoothed with a background language model based on collection term frequency estimates. The goal of the Lugano group was to highlight words from the term distribution of R2 that are rare in the term distribution of R1 in order to reduce the number of documents from RL1 that are present in RL3. To this end,term probabilities in R2 were weighted by their relative information in R2 and R1 to calculate a new query model R3: $p(w|R3) \propto p(w|R2) * log(p(w|R2)/p(w|R1))$. The normalizing constant in this case is simply the Kullback-Leibler divergence between R2 and R1. After obtaining the new term distribution, the top K terms were selected from R3 and submitted as a weighted query to Terrier again using the BM25 retrieval function. In this way two runs were generated by varying the effect of the background smoothing, `USIML052010` and `USIML092010`.

References