From *"Dango"* to *"Japanese Cakes"*: Query Reformulation Models and Patterns

Paolo Boldi[*], Francesco Bonchi[†], Carlos Castillo[†], Sebastiano Vigna[*]
[*]DSI, Università degli studi di Milano; Milan, Italy
[†]Yahoo! Research; Barcelona, Spain

Abstract

Understanding query reformulation patterns is a key step towards next generation web search engines: it can help improving users' web-search experience by predicting their intent, and thus helping them to locate information more effectively.

As a step in this direction, we build an accurate model for classifying user query reformulations into broad classes (generalization, specialization, error correction or parallel move), achieving 92% accuracy. We apply the model to automatically label two large query logs, creating annotated query-flow graphs. We study the resulting reformulation patterns, finding consistency with results from previous studies done on smaller manually annotated datasets, and discovering new interesting patterns, including connections between reformulation types and topical categories.

Finally, applying our findings to a third query log that is publicly available for research purposes, we demonstrate that the our reformulation classifier leads to improved recommendations in a query recommendation system.

1. Introduction

Information retrieval is an interactive and iterative process: only in approximately half of the cases an information need is satisfied with just a single query [1], [2]. In the other half of the cases, the user has to *reformulate* her initial query because it was over- or under-specified, or did not use terminology matching relevant documents, or simply contained errors or typos. The picture is made even more complex by the fact that, although queries are typically short [3], they usually form together chains of topically-related queries [4] sharing the same search goal. There is a general research trend in Web Information Retrieval towards trying to understand those tasks instead of looking at queries in isolation. The applications of this knowledge include building new or better search tools, such as query recommendation systems [5], [6] and improving the relevance of search engine results [7], among other goals.

The main source of information for understanding users' behavior and intent in web search are *query logs*. Extracting behavioral patterns from the wealth of information contained in query logs, is a key step to improve the service provided by search engines and to develop innovative web-search paradigms. In particular, and this is the focus of this paper, *by mining query logs we can understand the dynamics underlying the query reformulation process*, and use this knowledge in applications aimed at improving the user web-search experience.

In the following we process a query log first by applying the session segmentation model by Boldi *et al.* [8]. The output of this model are search missions: sequences of queries with a similar information need [4], [9].

Once search missions are determined, we focus on inferring the chain of reformulations underlying the search mission. First, for each query pair we want to determine which query reformulation type (abbreviated QRT) the user was doing. Second, from the whole search mission we want to extract the overall *strategy* followed by the user.

Our main contributions in this paper are:

1. Reformulation model. We show that accurate automatic classification of QRTs is possible. Learning automatically from a human-labeled query log sample, we build a model for automatic classification of QRTs (Section 4). Our model has a rather high accuracy, $\approx 92\%$ discriminating among four reformulation types. The classifier is able to predict correctly even very difficult cases. We describe in detail the process followed to build the model, and then we inspect the model behavior, e.g. with respect to the topical categories of the queries in the reformulation. To the best of our knowledge this is the first work learning a model for automatic classification of QRTs by mining a query log.

2. Reformulation strategies. Thanks to our model, we can automatically label very large query logs and analyze them (Section 6). We transform each search mission into a sequence of QRTs. Next, we find salient sequential patterns in these data, which represent high-level search strategies. We analyze tens of millions QRTs, finding consistency with previous studies done mostly over small, manually-assessed collections, and extending those results with new insights.

3. Reformulation graphs. Using our model we can annotate the arcs of a *Query Flow Graph* [8] with QRTs: we present a study on the properties of this annotated graph, including relationships between the various slices induced by the reformulation types. As an application of our approach, we study query recommendations based on short random walks on different slices of the query-flow graph. Our experiments show that our methods can match in precision, and often improve, query-click based recommendations without using click information. Our results also show that having QRT labels on the edges is crucial for obtaining high-quality recommendations.

Section 2 describes related work, and in Section 3 we discuss the taxonomy of QRTs that we adopt in this paper. Sections 4, 5, 6 and 7 present the model, the characterization, the sequential patterns, and the graphs obtained. In Section 8 we study an application to query recommendation, while in Section 9 we summarize our findings and discuss future work.

2. Related Work

Reformulation types. Although the classification of reformulation types is not a new research topic, this paper is the first attempt to automatically learn how to classify query-reformulation types (QRTs) by mining a query log, instead of providing a set of manually-crafted hard rules: the advantage of our approach stays in its generality (for example, it can be easily applied to other languages or to more specific domains).

The study of query reformulation types started with the work of Lau and Horvitz [10], who sampled 4 960 queries from a query log and manually labeled the transitions they found, proposing a classification of query-reformulation types (QRTs). Rieh and Xie [2] manually labeled 313 search missions and suggested a more fine-grained classification. While defining the classes of query reformulation types (Section 3) we have been mostly following their taxonomy.

Jansen *et al.* [11], [12] used manually-crafted rules to identify reformulation types (such as "adding one extra word means specializing the current query"). These rules follow concepts from [13], and they coincide perfectly with the definition of the classes, i.e., there's no automatic learning involved as in our work. They study patterns in a query-log containing 1.5 million reformulations.

Query graphs. The information extracted from query logs can be summarized and suitably represented through query graphs, several examples of which appear in Glance [14], Craswell and Szummer [15], Baeza-Yates and Tiberi [16]. In all such proposals, however, the notion of "clicked URL" plays a central role: for example, in [15] the query graph is bipartite, with nodes representing queries and documents, and with an arc connecting a query q and a document d iff d was clicked by some user after submitting the query q. Conversely, our model is intentionally simpler in a sense, because it does not use the clicked URLs.

Particularly relevant for this paper is the application of query log analysis to the segmentation of sessions into user missions or chains, introduced by Radlinski *et al.* [4]. Successful examples of such an application were presented by Jones and Klinkner [9]. In this paper we follow our previous work [8]: we use the same session segmentation model as a pre-processing step, and we we adopt the same graph representation of a query log, named *Query Flow Graph*, in which edges connect pairs of queries that appear consecutively in the query log, and are labelled with application-specific information.

Query-recommendations. Most of the work on query recommendation has focused on measures of query similarity [6], [17]: this is a prudent standpoint, but it often leads to unsurprising (although correct) recommendations. Baeza-Yates *et al.* [5] study the problem of suggesting related queries issued by other users and query expansion methods to construct artificial queries, whereas Fonseca *et al.* [17] discover related queries with a method based on association rules. Wen *et al.* [18] also present a clustering method for query recommendation based on different notions of distances, and Jones *et al.* introduced the notion of query substitution [19].

Craswell and Szummer [15] describe a method based on random walks on the query-click graph [20], that can be used to provide query recommendations as follows: given the input query, it computes the personalized PageRank [21] of all the other queries and then picks the top ones as recommendations. In Section 8 we use their method as a baseline when studying the application of our work to query-recommendations. Fuxman *et al.* [22] experiment with a similar approach in the context of finding related keywords for advertising. Mei, Zhou and Church [23] study a related system based on hitting time.

3. Defining Query Reformulation Types

Whenever the user enters two queries in sequence, we refer to the connection between the two queries as a *query transition*. If the user stays in the same search mission, we refer to this connection as a *query reformulation*. The goal of this work is to create a query reformulation model, and thus the first task is to define which are the target categories for the model. We adopt a taxonomy of query transitions inspired by Rieh and Xie [2], with some differences that we describe next.

Conceptually, our taxonomy has two dimensions, depicted in Figure 1. One dimension is found along the generalization-specialization axis, and the other dimension along the dissimilarity axis. As we move left to right along the dissimilarity (horizontal) axis, we find a continuous in which the syntactic and semantic gap between two queries gets larger and larger. As we move to the top or to the bottom along the specificity (vertical) axis, we find respectively reformulations towards more general or more specific queries.

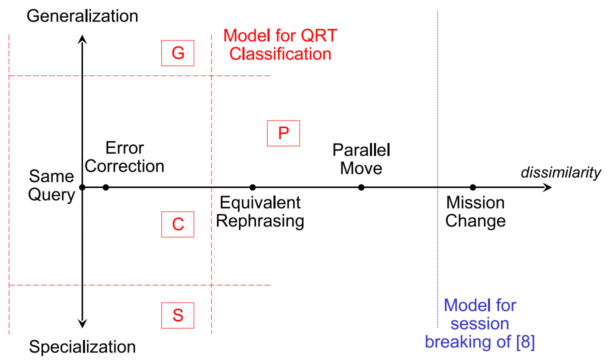


Fig. 1. Graphical depiction of transition types.

Dissimilarity axis. We discretize the dissimilarity axis as follows. We start with zero dissimilarity (*Same query*), moving on to *Error correction*: the user is trying a different spelling or capitalization of a query (e.g. "`califrnia`" and "`california`"). Next we find *Equivalent rephrasing*: changing the wording of the query, but keeping exactly the same goal, in the sense of [9] (e.g. "`used car`" and "`second-hand car`"). Then we find *Parallel move*: a modification of the query from one aspect of an entity to something related but not equivalent (e.g.: a "`hotel in Dublin`" and "`flights to Dublin`").

Finally, we have *mission change*: the user is completely changing topic and she is looking for something else [4], [9]. In the right side of Figure 1, the hyperplane separating *Mission change* from the rest represents the breaking of sessions into search missions. In our case we apply the model in [8] for this part.

Our classification of reformulations departs from the one proposed in [2] in the sense that they use a more fine-grained taxonomy (including classes such as parallel move, replacement with synonym, term variation, operator usage, type of resource and domain suffix). The work in [11], [12] presents a similar taxonomy, but they also distinguish between user-initiated reformulations and reformulations recommended by the search engine, and consider changes in the collection being queried, which in our case is always the Web. Both scenarios are outside the scope of the present paper.

Specificity axis. Along the vertical axis instead we have *Generalization* and *Specialization*. Generalization occurs when the new query q' is more general than q (e.g.: "`camping`" to "`outdoor activities`"); in some cases (but not all of them) a generalization can be automatically identified because q' is a conjunction with a proper subset of the terms of q; this type of rule is used in the manually-built classifier in [11]. In a specialization, instead, the new query q' is more specific than q (e.g. "`animal pictures`" and "`photos of African lions`").

We expect *Generalization* and *Specialization* to be related. A generalization reflects the user's desire to increase recall, whereas a specialization is the need to improve precision. We also expect some specific properties from these two transition types. For instance, we expect that both of them should define a transitive relation. Also, we expect that they should be anti-symmetric, given that, for instance, two queries can not be simultaneously a specialization of each other. Of course we do not expect these properties to hold deterministically given the noise present in the query log.

4. Learning a Query Reformulation Model

In this section we describe the process we followed in order to build a model for query-reformulation type classification.

Training data. We started from a set of *consecutive* query pairs (q, q'), sampled from a query log of the Yahoo! search engine in 2008 and segmented into search missions using the model of [8]. In order to create a training set for our QRT classification problem, a group of *editors* manually labelled the set of query pairs (q, q') in each search session with one of the reformulation types described in Section 3. In cases two or more editors disagree on the type of a query reformulation, the query pair was removed from the training set. This left us with a set of 1375 labelled examples, of which we used 2/3 for training and 1/3 for testing.

Features. We used a set of 27 features to build our model for QRT classification, including features from [8], [24], [13], [9] that have shown to be also effective for query segmentation.

TABLE 1. Description of the features extracted for each query reformulation (q, q').

Session-related features
[f1] Number of sessions in which reformulation (q, q') occurs; **[f2]** the same of [f1] divided by the number of sessions in which (q, x) occurs (for any x); **[f3]** the same of [f1] divided by the number of sessions in which (x, q') occurs (for any x); **[f4]** among all sessions containing (q, q'): average number of clicks since session begin, and **[f5]** since the query preceding (q, q'); **[f6]** average session size of other sessions containing (q, q'); **[f7]** average position in session expressed as number of queries before q since the session begun, and **[f8]** the ratio $f7/f6$; **[f9]** and **[f10]** fraction of occurrences in which this pair is the first (last) pair in the session.
Temporal features
[f11] average time elapsed between q and q' in each session in which both occur; **[f12]** sum of $1/t_i$ where t_i is the elapsed time between a query i and the previous event in a session.
Textual features
[f13] Levenshtein distance (a.k.a. edit distance); **[f14]** and **[f15]** length in characters of q and q' respectively; **[f16]** the difference $f15 - f14$; **[f17]** the ratio $f16/f14$. Then each query is turned into a bag of character tri-grams and we take **[f18]** the cosine similarity, **[f19]** the Jaccard coefficient, and **[f20]** the size of the intersection between the two bags. **[f21]**, **[f22]** and **[f23]** the same similarity measures but on stemmed terms instead of tri-grams. **[f24]** and **[f25]** number of terms in q and q'; **[f26]** the difference $f25 - f24$, and **[f27]** the ratio $f26/f24$.

For efficiency reasons, we used only features that consider the query sequences and the clicks of users, but that do not require access to the resulting URLs or page snippets. Although the latter information might be very powerful (or even decisive) to determine the query reformulation type, we wanted to limit ourselves to features that could be computed very quickly with little computational overhead. We note that all our features are available at run-time: for instance the "average session length" is the average over previously seen sessions containing a given query pair, not the session length of the current session which is unknown before the session ends. All the features passed a features selection phase in which we evaluated each feature relevance w.r.t. our target variable (i.e., query reformulation type). The features are presented in Table 1, and include session features (statistics relative to the sessions in which the pair (q, q') occurs, such as average session length, average position of the queries in the sessions etc.), temporal features (e.g., average time difference between q and q' in the sessions where (q, q') occurs) and textual features (textual similarity measures; some of them turn each query into a bag of words, and some into a bag of character trigrams).

Modelling. Standard methods such as boosted decision trees showed an accuracy of approximately 85% in predicting query reformulation types. The model that we built after trying several induction methods for our classification problem, exhibits an accuracy of 92% on a test set of unseen cases.

Instead of directly tackling the 4-classes problem, we built four distinct binary classification problems, where in each problem the target variable is being or not a certain QRT (e.g., $is\_G?$, $is\_S?$, etc.), plus a final 4-classes classifier for the undecided cases. Each of the five models is a rule-based classifier built with C5.0, the successor of the well-known C4.5 decision tree induction algorithm [25]. We placed the classifiers in cascade as in Figure 2, using a greedy method for optimizing the ordering of the individual classifiers.

The objective of the cascade is make some decisions with very high *precision* and put those cases aside. The rationale for this is that at this stage we do not care much about *false*

TABLE 2. Example of difficult cases in the test set classified correctly. Overall accuracy is 92%

q	*q′*	QRT
`dango`	`japanese cakes`	G
`Find somebody in Germany`	`Find my friend in berlin`	S
`Nutrition`	`Vegetarian Society`	S
`ikea`	`corner vanity units`	S
`sport`	`PSV Eindhoven v Tottenham`	S

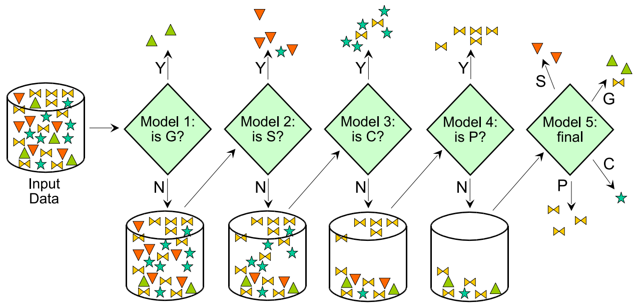


Fig. 2. Depiction of our QRT classification model.

negatives: they are not definitively errors, as they still have the chance to be classified correctly later by the fifth classifier. In order to boost precision (i.e., achieving very low number of false positives, while paying in terms of recall) while building the first 4 models we used the possibility of defining different *misclassification costs* for different kind of errors: e.g., telling to the classifier induction algorithm to weight a false positive the double of a false negative.

As examples pass through a classifier, not only the training set is reduced in number of examples, but it is also enriched in features. In fact, to each example that is predicted as negative, it is attached the confidence with which the classifier has done such a prediction.

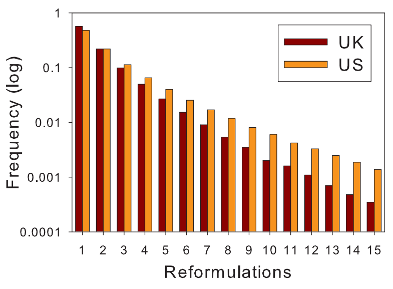
Our model is able to achieve a high accuracy also thanks to some very difficult prediction that it is able to do correctly. In Table 2 we report some of these difficult predictions.

Consider the example on the first row, that inspired the title of this paper: our classifier is able to correctly determine that the reformulation from "`dango`" to "`japanese cakes`" is a generalization, even in the absence of textual clues.

5. Empirical Study of Query Reformulations

Using our model we can automatically label query transitions in very large query logs to analyze typical reformulation patterns.

Datasets. We studied two large datasets from Yahoo! query logs. The first one corresponds to the searches in the Yahoo! UK search engine, from which the training data were extracted in the previous part. The second one corresponds to a completely different dataset from searches in the Yahoo! US search engine in 2008. All the queries from each user during a 1-month period are put together, and then segmented into search missions. Missions with a single query are discarded. Figure 3(a) reports the distribution of mission length in the two datasets.



(a)

	UK	US
G	4.4%	9.5%
S	37.5%	30.1%
C	10.4%	5.0%
P	47.7%	55.5%
	$n = 6M$	$n = 10M$

(b)

	US$\geq$ 5	Rie & Xie[2]
G	11.0%	12.7%
S	26.5%	23.4%
C	4.0%	5.2%
P	58.5%	56.2%
	$n = 4M$	$n = 2K$

(c)

Fig. 3. (a) distribution of the number of reformulations per search mission, in the two datasets. (b) and (c) distributions of QRTs (query-reformulation types).

The size of the datasets we analyze is much larger than those reported in the literature for this problem (313 chains with 5 QRTs each in [2], and about $1.5M$ reformulations in Jansen *et. al* [11], [12]): our UK and US datasets contain more than $3M$ and $4M$ search missions respectively (corresponding to $6.5M$ and $10.5M$ QRTs, resp.). Even when focusing only on search missions of length at least 5, we have $222K$ chains ($1.5M$ QRTs) in the UK dataset, and $527K$ chains ($4.3M$ QRTs) in the US dataset. In the following we denote **UK**≥ 5 and **US**≥ 5 the two datasets when we only consider long chains.

Query reformulation distribution. Each query reformulation in each mission was labeled with the model we described in Section 4. Figure 3(b) reports the distribution of reformulation types, which is consistent with the one of Rieh and Xie [2], considering the mapping between the query categories of each work and their minimum-length constraint. Also consistently with [2], the class `P` is largely the most populated (47%-58%). It is worth noting that this is slightly overestimated, as it is partially due to some *false negative* errors of the model used to segment sessions into chains: we have observed that *mission changes* that are not detected by that model are recognized as `P` (as it might be expected) by the model for QRT classification. On the generalization-specialization axis, specializations (30%-38%) are much more frequent than generalizations (4%-10%). This difference is however largely reduced when focussing on chains of length 5 or more, as shown in Figure 3(c).

Relationship with query topics. We conducted another experiment in order to assess how query reformulations and mission changes relate to query *topics*. There are many approaches to topical query classification, e.g. [26]. In this experiment we issued each query to our search engine, obtaining the top 20 documents, and we used an in-house automatic document classifier that maps them to the most likely Yahoo! Directory category for each document. Next we did a majority voting among the topics of the documents associated to the query, to determine the query topic. To increase precision at the expense of coverage, if the main topic was not at least twice as prevalent as the second topic we considered the query topic as "unknown". This is a slow yet very simple query classification

TABLE 3. Fraction of transitions where the top-level topic remains the same, and salient topic-transitions.

QRT	Topic match		Most salient topic transitions
G	UK	64%	1. reference→reference
			2. government→government
	US	64%	1. reference→government
			2. reference→reference
S	UK	59%	1. reference→reference
			2. government→ government
	US	71%	1. reference→reference
			2. government→ government
C	UK	54%	1. reference→computers and internet
			2. news and media→news and media
	US	53%	1. reference→health
			2. science→social science
P	UK	46%	1. arts→reference
			2. reference→government
	US	48%	1. reference→education
			2. social science→government
X	UK	22%	1. computers and internet→recreation
			2. entertainment→education
	US	23%	1. recreation→health
			2. soc. and culture→computers and internet

method that is nevertheless quite precise. We used it to classify by topic 100K queries from the UK data and 100K queries from the US data. For each query transition, we compared the top-level topic of the two queries involved in the transition: this is usually something very broad as “science → health”, etc. If the two topics coincide, we count this as a top-level topic match in Table 3. In the table we denote mission changes with the transition type X.

Obviously, whenever there is a mission change, the user is more likely to change the broad topic than to stay in the same broad topic. The opposite occurs in the case of generalization, specializations, and error corrections, in which the user is more likely to stay in the same broad topic. As expected, parallel moves are more ambiguous from the perspective of broad topics.

We verified whether some broad topics would more likely of motivating certain transition types than others. Table 3 shows some top-level topic pairs with the highest ratio of their probability conditioned to each transition type with respect to their prior probability. For generalization (G) and specialization (S), it is frequent to observe pairs of queries that are both reference search (dictionary/encyclopedia) or searching for some government-related topics. In the case of parallel moves (P), switches to and from reference search are common. As for mission change (X), we observe an interesting fact: there are frequent changes from and to recreation/entertainment topics which may signal alternating between work/study related activities and leisure.

Entropy of query reformulations. We looked at the entropy of the distribution of the next QRT among those that are observed on a query log, for a given query. To consider only queries for which we have enough information, we averaged the entropy over all queries having frequency larger than 100. We also looked at the extent to which the reformulation type is determined by the query. An average value close to 0 would mean that the query determines almost completely the reformulation type (for instance, that certain queries almost always are followed by a correction, while other queries almost always are followed by a parallel move, and so on). The actual value, shown in Table 4, is close to 1 meaning than when writing a reformulation for a query, the user will decide mostly between two reformulation types on average.

TABLE 4. Entropy measures

	UK data	US data
Reformulation-type entropy	1.1	1.0
Next-query entropy:		
Generalization (G)	1.0	1.3
Specialization (S)	5.4	2.6
Correction (C)	1.1	1.3
Parallel move (P)	6.5	4.0

We also examined the *next-query entropy* for a query q, given a reformulation type t. We averaged this over the same queries as with the reformulation-type entropy. The result is shown in Table 4. The next-query entropy is small for generalizations and error corrections, and close to 1 meaning that there is some variability: when generalizing or doing small changes in the query, users basically pick between 2 possible reformulated queries on average. The next-query entropy for specialization and parallel moves is substantially higher, from 3 to 6 bits, meaning that the users pick between several choices on average (the entropy may be lower in our US graph probably due to the removal of pairs with count equal to one).

6. Query Reformulation Strategies

In this section we present some observations about sequences of reformulations, which represent abstract *search strategies* followed by users. We start by transforming each search mission into a sequence of QRTs that we represent as strings with an X at the beginning and at the end, signaling the border of a search mission.

Conditional reformulation probability. In Table 5 we report the conditional probabilities of QRTs depending on the previous QRT. There are several insights from this table. Specializations are more likely to occur at the beginning of a chain; this means that many users starts with a broad query . Specializations are more likely to occur after a generalization, and conversely, the probability of a generalization is boosted after a specialization; so generalizations and specializations seems to be present in alternating order. Finally, error corrections are common at the beginning or end of a chain, or after another error correction.

TABLE 5. Ratio of the conditional probability of a QRT given the previous QRT, with respect to the priori probability.

	UK dataset					**US dataset**				
	Current					Current				
Next	G	S	C	P	X	G	S	C	P	X
G	0.8	**1.7**	**0.3**	**0.4**	1.2	0.6	**2.0**	0.6	0.6	0.9
S	1.3	0.7	**0.5**	0.7	**1.6**	1.4	0.6	0.6	0.7	**1.6**
C	**0.3**	**0.4**	1.2	0.6	**1.8**	**0.5**	**0.5**	**4.0**	0.7	**1.6**
P	**0.5**	0.9	0.6	0.8	1.4	0.6	0.8	0.7	1 .0	1.3
X	1.4	1.4	**1.7**	**1.5**	**0.0**	1.3	1.4	**1.5**	1.4	**0.0**

TABLE 6. Reformulation patterns with a frequency much higher than the expected one assuming independence.

	Frequency			
Pattern	**UK**	**US**	**UK**≥ 5	**US**≥ 5
XC	12.7%	5.6%	7.8%	4.5%
SG	2.8%	7.6%	16.4%	30.6%
GS	2.5%	6.1%	17.7%	30.3%
CX	11.3%	4.6%	6.1%	3.1%
XS	38.2%	35.5%	44.5%	34.5%
CC	1.4%	1.3%	5.1%	4.8%
SGS	0.9%	2.5%	8.6%	14.6%
CCC	0.3%	0.2%	1.5%	1.4%
GSG	0.2%	1.0%	2.5%	7.1%
SSG	0.7%	1.8%	7.6%	10.9%
XSG	1.7%	4.0%	4.1%	6.9%
SGX	1.3%	3.1%	2.2%	4.8%

Frequent reformulation patterns. We also looked for frequent reformulation sequences, in which frequency is the number of strings in the database containing a given pattern. We selected some patterns by means of an *interestingness* measure defined as the ratio between the real frequency and the *expected* frequency which is computed assuming independence of QRTs. Table 6 lists a few of the interesting patterns we found; they confirm and complement the findings in Table 5: error corrections are more frequent at the beginning of a chain, they also tend to appear contiguously, and specialization-generalization tend to appear in alternating order.

7. Annotated Query-flow Graph

The *Query Flow Graph* introduced by Boldi *et al.* [8] is a directed graph $G = (Q, E)$, where Q is the set of queries, $E \subseteq Q \times Q$ is the set of query transitions, and edges may hold application-dependent information. In our case we annotate the edges with two labels: a weight and the QRT as given by our model. Weights are defined as $w(q, q') = r(q, q')/\sum_{k:(q,k)\in E} r(q, k)$, where $r : E \to \mathbb{N}$ represents the number of times the transition was observed in the query log, so the weight $w(q, q')$ represents the probability of query q' following query q in a session. An small excerpt of this graph is shown in Figure 4.

We build two large query-flow graphs. For the UK dataset, we used all transitions, whereas for the US dataset, we discarded all hapax transitions (those with count one). The resulting graphs have the following sizes:

– UK: $21,247,414$ nodes, $21,216,958$ arcs (0.99 arcs/node);
– US: $58,312,610$ nodes, $53,960,925$ arcs (0.93 arcs/node).

We studied the graph by filtering according to the transition type; this way, each query-flow graph gave rise to five "slices" of the graph, one for each transition type. Table 7 presents some key statistics about each slice. All graphs are extremely sparse and essentially acyclic. If you delete from the graph all isolated nodes and isolated arcs (an arc (q, q') is isolated iff q has outdegree 1 and q' has outdegree 0), the fraction of remaining nodes is extremely small.

Anti-symmetry and correlations As explained in Section 3, some of the transition types should exhibit some natural properties; for example, both G and S should be anti-symmetric

TABLE 7. Basic properties of the query-flow graphs.

	Density arcs per node		SCC (largest)		Reciprocity $\rho(q, q', -)$	
	UK	US	UK	US	UK	US
G	0.04	0.06	0.00%	0.00%	0.0%	0.0%
S	0.31	0.26	0.25%	0.07%	0.2%	0.8%
C	0.07	0.05	0.14%	0.20%	1.7%	12.1%
P	0.41	0.25	2.51%	2.41%	1.6%	14.8%
X	0.17	0.39	1.10%	1.45%	3.1%	26.3%

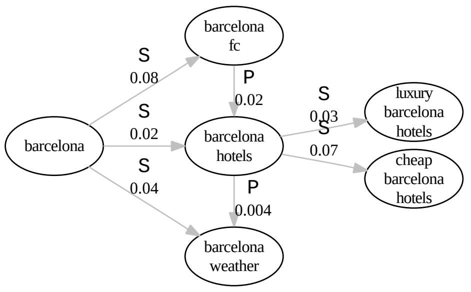


Fig. 4. Excerpt of the query-flow graph around the query "`barcelona hotels`" extracted from the UK dataset.

and transitive. Of course, we cannot expect these properties to hold deterministically, both because of the presence of noise and because we should take into account the frequency of each observed transition. A reasonable measure of symmetry is a **weighted reciprocity** that we define as follows: let $r(q, q', t)$ be the count associated with arc (q, q') in a given slice t, or zero if (q, q') is not an arc in t, and define $\rho(q, q', t) = \min(r(q, q', t), r(q', q, t))/ \max(r(q, q', t), r(q', q, t))$.

In the ideal case, if t defines a perfectly anti-symmetric relation this quantity should be 0 for all arcs in t, whereas it should be 1 for perfectly symmetric relations.

The average $\rho(q, q', -)$ for all arcs (q, q') is shown in Table 7: notice that the values are all very small, due to the sparsity of all graphs, but they are significantly closer to zero (or even exactly zero) for G and S, whereas they are several times larger for the other transition types.

8. Query Recommendation

Automatic detection of QRT is interesting from a web mining perspective but also has applications to improve web search. This section describes one such application (query recommendation) and summarizes an experiment comparing it to previous work. Further details can be found in [27].

Experiments. The experiments on query recommendations are done over a recent query-log dataset available for academics for research purposes: the "Spring 2006 Data Asset"[1] distributed by Microsoft Research. The data we used consist of a query log excerpt with 15 million queries, filtered to remove adult queries. Each record includes a query, an anonymous session-id, a timestamp, and the shown/clicked results. We processed the query log to create an annotated query-flow graph as in the previous sections. The query-flow graph is then sliced, and on each slice a random walk is performed

1. `http://research.microsoft.com/users/nickcr/wscd09/`

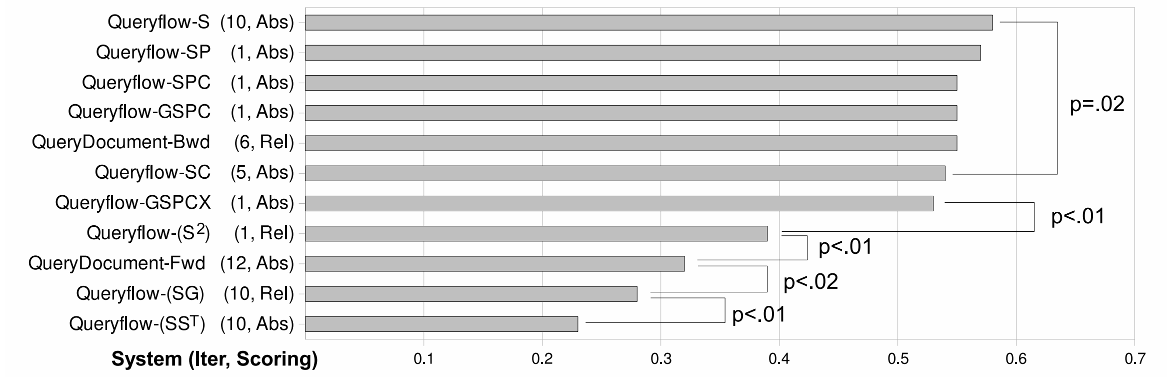


Fig. 5. Average precision@5 (useful or somewhat useful recommendations) per system.

starting from the input query, and using the resulting distribution for query recommendation in two different ways: in the *absolute* method, recommendations are sorted based on the probability they obtain in the random walk described; in the *relative* method, recommendations are sorted based on the ratio between the values obtained in the previous case and the PageRank values obtained by using no personalization (i.e., starting at random at any node). We perform the same experiments on transposed graph slices, unions of graph slices, or the composition[2] of two slices. We repeated the experiments with 1, 5 or 10 iteration steps (we omit the results obtained with more than 10 steps, as we observed no more improvement).

As a baseline, we implemented a graph-based query-recommendation system following the method by Craswell and Szummer [15], using both the "forward' and the "backwards" weighting schemes, with 6 or 12 iteration steps (an even number of steps is required, because their graphs are bipartite, and we need to end the random walk in a query and not in a document); we observed no improvements with larger numbers of iterations.

Assessment. The evaluation of the recommendations produced by the different systems was done as follows. A set of 114 input queries having frequencies between 700 and 15,000 was selected at random; we used these frequencies limit to avoid very frequent or infrequent queries. Test queries were very varied, e.g.: "`Grey’s anatomy`", "`Juno`", "`Maggie Gyllenhaal`', "`CNN news`", and "`guitar tabs`"; we discarded all the queries containing a domain name. Then we generated an evaluation pool with the union of the top 5 recommendations from each system; this yielded on average 53.4 different recommendations per query.

Next, a group of 5 assessors were asked to rate each recommendation as **useful**, **somewhat useful** or **not useful**. From the assessments, 62% of the recommendations were considered not useful, 12% somewhat useful, and 25% useful; for the remaining 1% the editors could not assess due to non-English queries or other reasons.

2. By "composition" of two graphs with the same node set we mean the multiplication of the respective adjacency matrices in the semiring having multiplication as sum and maximization as product.

Results. Figure 5 contains a chart of the best performing variant of each system. The average precision@5, is the probability that a top-5 recommendation issued by a system is labeled as "useful" or "somewhat useful". The significance p of the difference between two systems is the probability of observing for the second system the obtained score or lower by chance, assuming that both systems have the same accuracy. Only the best-performing parameter setting for each system is shown in the figure in parenthesis as a pair (number of iterations, scoring method). Notice that we are here testing our systems against a very strong null hypothesis, because only the top 5 recommendations are being considered, and many of them are correct; so the probability of guessing *among them* is very high, even at random.

We observe that the usefulness of the recommendations decreases as we introduce more transition types: specialization transitions seem to produce the most useful recommendations (Queryflow-S), whereas adding parallel moves (Queryflow-SP), corrections (Queryflow-SPC), and eventually generalization (Queryflow-GSPC) or even transitions that are not part of the same mission (Queryflow-GSPCX) results in less useful recommendations. This means that being able to discriminate among QRTs is key to produce better recommendations.

The "absolute" scoring method is usually better than the "relative" scoring method, and doing multiple iterations is often better than doing just one. The recommendations based on the baseline have either the same performance as recommendations using Queryflow-S, or a lower performance, at a significance of $p = 0.07$ (significance not drawn in the chart).

9. Conclusions

Main findings. During the course of this research, we have found that it is possible to automatically determine the type of a query reformulation, if the appropriate features are used. We have achieved 92% accuracy in distinguishing among 4 main classes of query reformulations, noticing that the learning scheme is important as it can exploit the fact that some class boundaries are more fuzzy than others. We applied the classifier to a large query log and studied reformulation paths that are the sequences of reformulations that a user does in the course of a search mission. This allowed us to study query reformulation patterns, matching some results of previous studies done over much smaller data set using manual assessments, and extracting new patterns which are discoverable giving that our automatic classifier enables the processing of a massive amount of data. From some of the patterns we extracted, we can see for instance that generalization and specializations appear frequently together in alternating order, and that error corrections are more frequent either at the beginning of a search mission or after another error correction. When mapping query transitions to topical categories we see that reference search is a typical context for generalizations and specializations, and that many mission changes are associated to switches from or to entertainment/recreation topics.

We annotated a large query flow graph with transition types, finding the anti-symmetry of generalization and specialization there. We also observed that given a query, the distribution of possible generalizations and error corrections tend to be more concentrated than the distributions of specializations or parallel moves.

One of the possible applications of the automatic classification of transition type is to generate query recommendations. In our experiments, we matched (and often improved) the quality of recommendations obtained using query-click graphs without using clicks. This means that the information contained in the annotated query-flow graph about consecutive queries in a session is as useful for this task as the user's clicks. Given that both data sources are independent, recommendations produced by a composition of both methods are worth to be investigated as future work.

Future work. There are several applications of our work to off-line query-log analysis. A promising application is graph-regularized automatic query classification. Such a system would start with a query classifier and a taxonomy of topics, and incorporate constraints of the type "if query q' is a specialization of query q, then the topic of query q' should be a descendant of the topic of query q". Another application is the diversification of search engine results. Popular specializations of a query, obtained automatically from query logs, represent different aspects of the original query that can be used to help the search engine include pages covering different user intents.

Finally, simultaneously learning both the query reformulation types and how to segment a session into chains (the two tasks that we identified and separated in the Introduction) might be a way of achieving a non-trivial improvement in accuracy. This would mean formulating our task in similar terms as, for instance, the task of *part-of-speech* and *bracketing* in Natural Language Processing. Also the insights obtained from the analysis of the graph can be used, e.g. by imposing an asymmetry constraint between specialization and generalization during the learning process.

References

[1] A. Spink, B. J. Jansen, D. Wolfram, and T. Saracevic, "From e-sex to e-commerce: Web search changes," *IEEE Computer*, vol. 35, no. 3, 2002.

[2] S. Y. Rieh and H. Xie, "Analysis of multiple query reformulations on the web: the interactive information retrieval context," *Inf. Process. Manage.*, vol. 42, no. 3, pp. 751–768, 2006.

[3] N. J. Belkin, "The human element: helping people find what they don't know," *Commun. ACM*, vol. 43, no. 8, 2000.

[4] F. Radlinski and T. Joachims, "Query chains: learning to rank from implicit feedback," in *Proc. of the 11th ACM SIGKDD int. conf. on Knowledge discovery in data mining* (KDD'05).

[5] R. Baeza-yates, C. Hurtado, and M. Mendoza, "Query recommendation using query logs in search engines," in *In International Workshop on Clustering Information over the Web* (ClustWeb, 2004).

[6] Z. Zhang and O. Nasraoui, "Mining search engine query logs for query recommendation," in *Proc. of the 15th int. conf. on World Wide Web* (WWW'06).

[7] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey, "An experimental comparison of click position-bias models," in *Proc. of the int. conf. on Web Search and Data Mining* (WSDM'08).

[8] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The query-flow graph: Model and applications," in *Proc.of the ACM 17th Conf. on Information and Knowledge Management* (CIKM'08).

[9] R. Jones and K. L. Klinkner, "Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs," in *Proc.of the ACM 17th Conf. on Information and Knowledge Management* (CIKM'08).

[10] T. Lau and E. Horvitz, "Patterns of search: analyzing and modeling web query refinement," in *Proc. of the 7th int. conf. on User modeling* (UM'99).

[11] B. J. Jansen, A. Spink, and B. Narayan, "Query modifications patterns during web searching," in *Proc. of 4th int. conf. on Information Technology* (ITNG'07).

[12] B. J. Jansen, M. Zhang, and A. Spink, "Patterns and transitions of query reformulation during web searching," *International Journal of Web Information Systems*, vol. 3, no. 4, pp. 328–340, 2007.

[13] D. He, A. Göker, and D. J. Harper, "Combining evidence for automatic web session identification," *Inf. Process. Manage.*, vol. 38, no. 5, pp. 727–742, September 2002.

[14] N. S. Glance, "Community search assistant," in *In Artificial Intelligence for Web Search*, 2001, pp. 91–96.

[15] N. Craswell and M. Szummer, "Random walks on the click graph," in *Proc. of the 30th int. ACM SIGIR conf. on Research and development in information retrieval* (SIGIR'07).

[16] R. Baeza-Yates and A. Tiberi, "Extracting semantic relations from query logs," in *Proc. of the 13th ACM SIGKDD int. conf. on Knowledge discovery and data mining* (KDD'07).

[17] B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani, "Using association rules to discover search engines related queries," in *Proc. of the 1st Latin American Web Congress* (LA-WEB'03).

[18] J.-R. Wen, J.-Y. Nie, H.-J. Zhang, and H.-J. Zhang, "Clustering user queries of a search engine," in *Proceedings of the 10th int. conf. on World Wide Web* (WWW'01).

[19] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating query substitutions," in *Proc. of the 15th int. conf. on World Wide Web* (WWW'06).

[20] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in *Proc. of the 6th ACM SIGKDD int. conf. on Knowledge discovery and data mining* (KDD'00).

[21] G. Jeh and J. Widom, "Scaling personalized web search," in *Proc. of the 12th int. conf. on World Wide Web* (WWW'03).

[22] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal, "Using the wisdom of the crowds for keyword generation," in *Proc. of the 17th int. conf. on World Wide Web* (WWW'08).

[23] Q. Mei, D. Zhou, and K. Church, "Query suggestion using hitting time," in *Proc.of the ACM 17th Conf. on Information and Knowledge Management* (CIKM'08).

[24] D. He and A. Göker, "Detecting session boundaries from web user logs," in *Proc. of the BCS-IRSG 22nd annual colloquium on information retrieval research*.

[25] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[26] Y. Li, Z. Zheng, and H. . Dai, "Kdd cup-2005 report: facing a great challenge," *SIGKDD Explor. Newsl.*, vol. 7, no. 2, pp. 91–99, December 2005.

[27] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna, "Query suggestions using query-flow graphs," in *Proc. of the 2009 workshop on Web Search Click Data* (WSCD'09).