Identifying Web Search Query Reformulation using Concept based Matching

Ahmed Hassan
Microsoft Research
One Microsoft Way
Redmond, WA 98053, USA
hassanam@microsoft.com

Abstract

Web search users frequently modify their queries in hope of receiving better results. This process is referred to as "*Query Reformulation*". Previous research has mainly focused on proposing query reformulations in the form of suggested queries for users. Some research has studied the problem of predicting whether the current query is a reformulation of the previous query or not. However, this work has been limited to bag-of-words models where the main signals being used are word overlap, character level edit distance and word level edit distance. In this work, we show that relying solely on surface level text similarity results in many false positives where queries with different intents yet similar topics are mistakenly predicted as query reformulations. We propose a new representation for Web search queries based on identifying the concepts in queries and show that we can significantly improve query reformulation performance using features of query concepts.

1 Introduction

Web search is a process of querying, learning and reformulating queries to satisfy certain information needs. When a user submits a search query, the search engine attempts to return the best results to that query. Oftentimes, users modify their search queries in hope of getting better results. Typical search users have low tolerance to viewing lowly ranked search results and they prefer to reformulate the query rather than wade through result listings (Jansen and Spink, 2006). Previous studies have also shown that 37% of search queries are reformulations to previous queries (Jansen et al., 2007) and that 52% of users reformulate their queries (Jansen et al., 2005).

Understanding query reformulation behavior and being able to accurately identify reformulation queries have several benefits. One of these benefits is learning from user behavior to better suggest automatic query refinements or query alterations. Another benefit is using query reformulation prediction to identify boundaries between search tasks and hence segmenting user activities into topically coherent units. Also, if we are able to accurately identify query reformulations, then we will be in a better position to evaluate the satisfaction of users with query results. For example, search satisfaction is typically evaluated using clickthrough information by assuming that if a user clicks on a result, and possibly dwells for a certain amount of time, then the user is satisfied. Identifying query reformulation can be very useful for finding cases where the users are not satisfied even after a click on a result that may have seemed relevant given its title and summary but then turned out to be not relevant to the user's information need.

Previous work on query reformulation has either focused on automatic query refinement by the search system, e.g. (Jones et al., 2006; Boldi et al., 2008) or on defining taxonomies for query reformulation strategies, e.g. (Lau and Horvitz, 1999; Anick, 2003). Other work has proposed solutions for the query reformulation prediction problem or for the similar problem of task boundary identification (Radlinski and Joachims, 2005; Jones and Klinkner, 2008). These solutions have adopted the bag-of-words approach for representing queries and mostly used features of word overlap or character and word level edit distances. Take the queries "hotels in New York City" and "weather in New York City" as an example. The two queries are very likely to have been issued by a user who is planning to travel to New York City. The two queries have 5 words each, 4 of them are shared by the two queries. Hence, most of the solutions proposed in previous work for this problem will incorrectly assume that the second query is a reformulation of the first due to the high word overlap ratio and the small edit distance. In this work, we propose a method that goes beyond the bag-of-words method by identifying the concepts underlying these queries. In the previous example, we would like our method to realize that in the first query, the user is searching for "hotels" while for in the second query, she is searching for the "weather" in New York City. Hence, despite similar in terms of shared terms, the two queries have different intents and are not reformulations of one another.

To this end, we conducted a study where we collected thousands of consecutive queries and trained judges to label them as either reformulations or not. We then built a classifier to identify query reformulation pairs and showed that the proposed classifier outperforms the state-of-the-art methods on identifying query reformulations. The proposed method significantly reduces false positives (non-reformulation pairs incorrectly classified as reformulation) while achieving high recall and precision.

2 Related Work

There are three areas of work related to the research presented in this paper: (i) query reformulation taxonomies, (ii) automatic query refinement, and (iii) search tasks boundary identification. We cover each of these areas in turn.

2.1 Query Reformulation Taxonomies

Existing research has studied how web search engines can propose reformulations, but has given less attention to how people perform query reformulations. Most of the research on manual query reformulation has focused on building taxonomies of query reformulation. These taxonomies are generally constructed by examining a small set of query logs. Anick (2003) classified a random sample of 100 reformulations by hand into eleven categories. Jensen et al. (2007) identified 6 different kinds of reformulation states (New, Assistance, Content Change, Generalization, Reformulation, and Specialization) and provided heuristics for identifying them. They also used them to predict when a user is most receptive to automatic query suggestions. The same categories were used in several other studies (Guo et al., 2008; Lau and Horvitz, 1999). Huang and Efthimis (2010) proposed another reformulation taxonomy. Their taxonomy was lexical in nature (e.g., word reorder, adding words, removing words, etc.). They also proposed the use of regular expressions to identify them. While studying re-finding behavior, Teevan et al. (2006) constructed a taxonomy of query re-finding by manually examining query logs, and implemented algorithms to identify repeat queries, equal click queries and overlapping click queries. None of this work has built an automatic classifier distinguishing reformulation queries from other queries. Heuristics and regular expressions have been used in (Huang et al., 2010) and (Jansen et al., 2007) to identify different types of reformulations. This line of work is relevant to our work because it studies query reformulation strategies. Our work is different because we build a machine-learned predictive model to identify query reformulation while this line of work mainly focuses on defining taxonomies for reformulation strategies.

2.2 Automatic Query Refinement

A close problem that has received most of the research attention in this area is the problem of automatically generating query refinements. These refinements are typically offered as query suggestions to the users or used to alter the user query before submitting it to the search engine.

Boldi et al. (2008) introduced the concept of the query-flow graph where every query is represented by a node and edges connect queries if it is likely for users to move from one query to another. Mei et al. (2008) used random walks over a bipartite graph of queries and URLs to find query refinements. Query logs were used to suggest query refinements in (Baeza-Yates et al., 2005). Hierarchical agglomerative clustering was used to group similar queries that can be used as suggestions for one another. Other research has adopted methods based on query expansion (Mitra et al., 1998) or query substitution (Jones et al., 2006). This line of work is different from our work because it focuses on automatically generating query refinements while this work focuses on identifying cases of manual query reformulations.

2.3 Search Task Boundary Identification

The problem of classifying the boundaries of the user search tasks within sessions in web search logs has been widely addressed before. This problem is closely related to the problem of identifying query reformulation. A search task has been defined in (Jones and Klinkner, 2008) as a single information need that may result in one or more queries. Similarly , Jansen et al. (2007) defined a session as a series of interactions by the user toward addressing a single information need. On the other hand, a query reformulation is intended to modify a previous query in hope of getting better results to satisfy the same information need. From these definitions, it is clear how query reformulation and task boundary detection are two sides of the same problem.

Boldi et al. (2008) presented the concept of the query-flow graph. A query-flow graph represents chains of related queries in query logs. They use this model for finding logical session boundaries and query recommendation. Ozmutlu (2006) proposed a method for identifying new topics in search logs. He demonstrated that time interval, search pattern and position of a query in a user session, are effective for shifting to a new topic. Radlinski and Joachims (2005) study sequences of related queries (query chains). They used that to generate new types of preference judgments from search engine logs to learn better ranked retrieval functions.

Arlitt (2000) found session boundaries using a calculated timeout threshold. Murray et al. (2006) extended this work by using hierarchical clustering to find better timeout values to detect session boundaries. Jones and Klinkner (2008) also addressed the problem of classifying the boundaries of the goals and missions in search logs. They showed that using features like edit distance and common words achieves considerably better results compared to timeouts. Lucchese et al. (Lucchese et al., 20011) uses a similar set of features as (Jones and Klinkner, 2008), but uses clustering to group queries in the same task together as opposed to identifying task boundary as in (Jones and Klinkner, 2008). This line of work is perhaps the closest to our work. Our work is different because it goes beyond the bag of words approach and tries to assess query similarity based on the concepts represented in each query. We compare our work to the state-of-the-art work in this area later in this paper.

3 Problem Definition

We start by defining some terms that will be used throughout the paper:

Definition: *Query Reformulation* is the act of submitting a query Q_2 to modify a previous search query Q_1 in hope of retrieving better results to satisfy the same information need.

Definition: A *Search Session* is group of queries and clicks demarcated with a 30-minute inactivity timeout, such as that used in previous work (Downey et al., 2007; Radlinski and Joachims, 2005).

Search engines receive streams of queries from users. In response to each query, the engine returns a set of search results. Depending on these results, the user may decide to click on one or more results, submit another query, or end the search session. In this work, we focus on cases where the user submits another query. Our objective is to solve the following problem: Given a query Q_1, and the following query Q_2, predict whether Q_2 is reformulation of Q_1.

4 Approach

In this section, we propose methods for predicting whether the current query has been issued by the user to reformulate the previous query.

4.1 Query Normalization

We perform standard normalization where we replace all letters with their corresponding lower case representation. We also replace all runs of whitespace characters with a single space and remove any leading or trailing spaces. In addition to the standard normalization, we also break queries that do not respect word boundaries into words. Word breaking is a well-studied topic that has proved to be

Table 1 : Examples of queries, the corresponding segmentation, and the concept representation. Phrases are separated by “\|” and different tokens in a keyword are separated by “_”

Query	Phrases and Keywords	Concept Representation
hotels in new york city	hotels in new_york_city	Concept1 {head=“hotels”, modifiers = “new york city”}
hyundai roadside assistance phone number	hyundai roadside_assistance | phone_number	Concept1 {head = “roadside _assistance”, modifiers = “hyundai”}, Concept2{“phone_number”}
kodak easyshare recharger chord	kodak_easyshare recharger_cord	Concept1{head =“recharger_cord”, modifiers =“kodak easyshare”}
user reviews for apple iphone	user_reviews for apple_iphone	Concept1{head=“user_reviews” , modifiers = “apple iphone”}
user reviews for apple ipad	user_reviews for apple_ipad	Concept1{head =“user_reviews”, modifiers = “apple ipad”}
tommy bhama rug	tommy_bhama rug	Concept1{head =“rug”, modifiers =“tommy bhama”}
tommy bhama perfume	tommy_bhama perfume	Concept1{head =”perfume”, modifiers =“tommy bhama”}

useful for many natural language processing applications. This becomes a frequent problems with queries when users do not observe the correct word boundaries (for example: “southjerseycraigslist” for “south jersey craiglist”) or when users are searching for a part of a URL (for example “quincycollege” for “quincy college”). We used a freely available word breaker Web service that has been described at (Wang et al., 2011).

4.2 Queries to Concepts

Lexical similarity between queries has been often used to identify related queries (Jansen et al., 2007). The problem with lexical similarity is that it introduces many false negatives (e.g. synonyms) , but this can be handled by other features as we will describe later. More seriously, it introduces many false positives. Take the following query pair as an example Q_1: weather in new york city and Q_2: “hotels in new york city”. Out of 5 words, 4 words are shared between Q_1 and Q_2. Hence, any lexical similarity feature would predict that the user submitted Q_2 as a reformulation of Q_1. What we would like to do is to have a query representation that recognizes the difference between Q_1 and Q_2.

If we look closely at the two queries, we will notice that in the first query, the user is looking for the “weather”, while in the second query the user is looking for “hotels”. We would like to recognize “weather”, and “hotels” as the head keywords of Q_1 and Q_2 respectively, while “new york city” is a modifier of the head keyword in both cases. To build such a representation, we start by segmenting each query into phrases. Query segmentation is the process of taking a users search query and dividing the tokens into individual phrases or semantic units (Bergsma and Wang, 2007). Many approaches to query segmentation have been presented in recent research. Some of them pose the problem as a supervised learning problem (Bergsma and Wang, 2007; Yu and Shi, 2009). Many of the supervised methods though use expensive features that are difficult to re-implement.

On the other hand, many unsupervised methods for query segmentation have also been proposed (Hagen et al., 2011; Hagen et al., 2010). Most of these methods use only raw web n-gram frequencies and are very easy to re-implement. Additionally, Hagen et al. (2010) have shown that these methods can achieve segmentation accuracy comparable to current state-of-the-art techniques using supervised learning. We opt for the unsupervised techniques to perform query segmentation. More specifically, we adopt the mutual information method (MI) used throughout the literature. A segmentation for a query is obtained by computing the pointwise mutual information score for each pair of consecutive words. More formally, for a query $x = \{x_1, x_2, ..., x_n\}$

$$PMI(x_i, x_{i+1}) = log\frac{p(x_i, x_{i+1})}{p(x_i)p(x_{i+1})} \quad (1)$$

where $p(x_i, x_{i+1})$ is the joint probability of occurrence of the bigram (x_i, x_{i+1}) and $p(x_i)$ and $p(x_{i+1})$ are the individual occurrence probabilities of the two tokens x_i and x_{i+1} .

A segment break is introduced whenever the point wise mutual information between two consecutive words drops below a certain threshold τ. The threshold we used, $\tau = 0.895$, was selected to maximize the break accuracy (Jones et al., 2006) on the Bergsma-Wang-Corpus (Bergsma and Wang, 2007). Furthermore, we do not allow a break to happen between a noun and a proposition (e.g. no break can be introduced between “hotels” and “in” or “in” and “new York” in the query “hotels in new york city”). We will shorty explain how we obtained the part-of-speech tags.

In addition to breaking the query into phrases, we were also interested in grouping multi-word keywords together (e.g. “new york”, “Michael Jackson”, etc.). The intuition behind that is that a query containing the keyword “new york” and another containing the keyword “new mexico” should not be awarded because they share the word “new”. We do that by adopting a hierarchical segmentation technique where the same segmentation method described above is reapplied to every resulting phrase with a new threshold $\tau_s < \tau$. We selected the new threshold, $\tau = 1.91$, to maximize the break accuracy over a set of a random sample of 10,000 Wikipedia title of persons, cities, countries and organizations and a random sample of bigrams and trigrams from Wikipedia text.

In our implementation, the probabilities for all words and n-grams have been computed using the freely available Microsoft Web N-Gram Service (Huang et al., 2010).

Now that we have the phrases and keywords in each query, we assume that every phrase corresponds to a semantic unit. Every semantic unit has a head and a zero or more modifiers. Dependency parsing could be used to identify the head and modifiers from every phrase. However, because queries are typical short and not always well-formed sentences, this may pose a challenge to the dependency parser. But as we are mainly interested in short noun phrases, we can apply a simple set of rules to identify the head keyword of each phrase using the part of speech tags of the words in the phrase. A part-of-speech (POS) tagger assigns parts of speech to each word in an input text, such as noun, verb, adjective, etc. We used the Stanford POS tagger, using Stanford CoreNLP, to assign POS tags to queries (Toutanova et al., 2003). To identify the head and attributes of every noun phrase, we use the following rules:

- For phrases with of the form: “NNX+” (i.e. one more nouns, where NNX could be NN: noun, singular, NNS: noun, plural, NNP: proper noun, singular or NNPS: proper noun, plural), the head is the last noun keyword and all other keywords are treated as attributes/modifiers.
- For the phrases of the form “NNX+ IN NNX+”, where IN denotes a preposition or a subordinating conjunction (e.g. “in”, “of”, etc.), the head is the last noun keyword before the preposition.

Table 1 shows different examples of queries, the corresponding phrases, keywords, and concepts. For example the query “kodak easyshare recharger chord” consists of a single semantic unit (phrase) and two keywords “Kodak easyshare” and “recharger cord”. The head of this semantic unit is the keyword “recharger cord” and “kodak easyshare” is regarded as an attribute/modifier. Another example is the two queries “tommy bhama rug” and “tommy bhama perfume”. The head of the former is “rug”, while the head of the latter is “perfume”. Both share the attribute “tommy bhama”. This shows that the user had two different intents even though most of the words in the two queries are shared.

4.3 Matching Concepts

Phrases in two concepts may have full term overlap, partial term overlap, or no direct overlap yet are semantically similar. To capture concept similarity, we define four different ways of matching concepts ranked from the most to the least strict:

- Exact Match: The head and the attributes of the two concepts match exactly.
- Approximate Match: To capture spelling variants and misspelling, we allow two keywords to match if the Levenshtein edit distance between them is less than 2.
- Lemma Match: Lemmatization is the process of reducing an inflected spelling to its lexical root or lemma form. We match two concepts if the lemmas of their keywords can be matched.
- Semantic Match: We compute the concept similarity by measuring the semantic similarity between the two phrases from which the concepts where extracted. Let $Q = \{q_1, ..., q_I\}$ be one phrase and $S = \{s_1, ..., s_J\}$ be another, the semantic similarity between these two phrases can be measured by estimating the probability of one of them being a translation of another. The translation probabilities can be estimated using the IBM Model 1 (Brown et al., 1993; Berger and Lafferty, 1999). The model was originally proposed to model the probability of translating from one sequence of words in one language to another. It has been also used in different IR applications to estimate the probability of translating from one sequence of words to another sequence in the same language (e.g. (Gao et al., 2012), (Gao et al., 2010) and (White et al., 2013)). More formally, the similarity between two sequences of words, $Q = \{q_1, ..., q_I\}$ and $S = \{s_1, ..., s_J\}$, can be defined as:

$$P(S|Q) = \prod_{i=1}^{I} \sum_{j=1}^{J} P(s_i|q_j)P(q_j|Q) \qquad (2)$$

where $P(q|Q)$ is the unigram probability of word q in query Q. The word translation probabilities $P(s|q)$ are estimated using the query-title pairs derived from the clickthrough search logs, assuming that the title terms are likely to be the desired alternation of the paired query. The word translation probabilities $P(s|q)$ (i.e. the model parameters θ) are optimized by maximizing the probability of generating document titles from queries over the entire training corpus:

$$\theta^* = argmax_\theta \prod_{i=1}^{M} P(S|Q, \theta) \qquad (3)$$

where $P(S|Q, \theta)$ is defined as:

$$P(S|Q, \theta) = \frac{\epsilon}{(J+!)^I} \prod_{i=1}^{I} \sum_{j=1}^{J} P(s_i|q_j) \qquad (4)$$

where ϵ is a constant, I is the token length of S, and J is the token length of Q. The query-title pairs used for model training are sampled from one year worth of search logs from a commercial search engine. The search logs do not intersect with the search logs where the data described in Section5.1 has been sampled from. S and Q are considered a match if $P(S|Q, \theta) > 0.5$.

4.4 Features

4.4.1 Textual Features

Jones and Klinkner (2008) showed that word and character edit features are very useful for identifying same task queries. The intuition behind this is that consecutive queries which have many words and/or characters in common tend to be related. The features they used are:

- normalized Levenshtein edit distance
- 1 if $lev > 2$, 0 otherwise
- Number of characters in common starting from the left
- Number of characters in common starting from the right
- Number of words in common starting from the left
- Number of words in common starting from the right
- Number of words in common
- Jaccard distance between sets of words

4.4.2 Concept Features

As we explained earlier the word and character edit features capture similarity between many pairs of queries. However, they also tend to mis-classify many other pairs especially when the two queries share many words yet have different intents. We used the conceptual representation of queries described in the previous subsection to compute the following set of features, notice that every feature has two variants one at the concept level and the other at the keyword (head or attribute) level:

- Number of "exact match" concepts in common
- Number of "approximate match" concepts in common
- Number of "lemma match" concepts in common
- Number of "semantic match" concepts in common
- Number of concepts in Q_1
- Number of concepts in Q_2
- Number of concepts in Q_1 but not in Q_2
- Number of concepts in Q_1 but not in Q_2
- 1 if Q_1 contains all Q_2s concepts
- 1 if Q_2 contains all Q_1s concepts
- all above features recomputed for keywords instead of concepts

4.4.3 Other Features

Other features, that have been also used in (Jones and Klinkner, 2008), include temporal features:

- time between queries in seconds
- time between queries as a binary feature (5 mins, 10 mins, 20 mins, 30 mins, 60 mins, 120 mins)

and search results feature:

- cosine distance between vectors derived from the first 10 search results for the query terms.

4.5 Predicting Reformulation Type

There are different strategies users use to reformulate a query which results in different types of query reformulations:

- Generalization: A *generalization* reformulation occurs when the second query is intended to seek more general information compared to the first query
- Specification: A *specification* reformulation occurs when the second query is intended to seek more specific information compared to the first query
- Spelling: A *spelling* reformulation occurs when the second query is intended to correct one or more misspelled words in the first query
- Same Intent: A *same intent* reformulation occurs when the second query is intended to express the same intent as the first query. This can be the result of word substitution or word reorder.

We used the following features to predict the query reformulation type:

- Length (num. characters and num. words) of Q_1, Q_2 and difference between them
- Number of out-of-vocabulary words in Q_1, Q_2 and the difference between them
- num. of "exact match" concepts in common
- num. of "approximate match" concepts in common
- num. of "lemma match" concepts in common
- num. of "semantic match" concepts in common
- num. of concepts in Q_1, Q_2 and the difference between them
- num. of concepts in Q_1 but not in Q_2
- num. of concepts in Q_1 but not in Q_2
- 1 if Q_1 contains all Q_2s concepts
- 1 if Q_2 contains all Q_1s concepts
- all concept features above recomputed for keywords instead of concepts

5 Experiments and Results

5.1 Data

Our data consists of query pairs randomly sampled from the queries submitted to a commercial search engine during a week in mid-2012. Every record in our data consisted of a consecutive query pair (Q_i,Q_{i+1}) submitted to the search engine by the same user and in the same session (i.e. within less than 30 minutes of idle time, the 30 minutes threshold has been frequently used in previous work, e.g. (White and Drucker, 2007)). Identical queries were excluded from the data because they are always labeled as reformulation and their label is very easy to predict. Hence, when included, they result in unrealistically high estimates of the performance of the proposed methods. All data in the session to which the sampled query pair belongs were recorded. In addition to queries, the data contained a timestamp for each page view, all elements shown in response to that query (e.g. Web results, answers, etc.), and visited Web page or clicked answers. Intranet and secure URL visits were excluded. Any personally identifiable information was removed from the data prior to analysis.

Annotators were instructed to exhaustively examine each session and "re-enact" the user's experience. The annotators inspected the entire search results page for each of Q_i and Q_{i+1}, including URLs, page titles, relevant snippets, and other features. They were also shown clicks to aid them in their judgments. Additionally, they were also shown queries and clicks before and after the query pair of interest. They were asked to then use their assessment of the user's objectives to determine whether Q_{i+1} is a reformulation of Q_i. Each query pair was labeled by three judges and the majority vote among judges was used. Because the number of positive instances is much smaller than the number of negative instances, we used all positive instances and an equal number of randomly selected negative instances leaving us with approximately 6000 query pairs.

Judges were also asked to classify reformulations into one of four different categories: Generalization (second query is intended to seek more general information), Specification (second query is intended to seek more specific information), Spelling (second query is intended to correct spelling mistakes), and Same Intent (second query is intended to express the same intent in a different way).

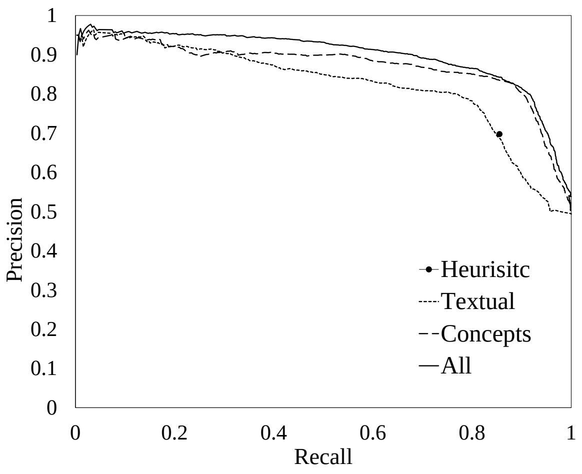

Figure 1 Precision-Recall Curves for the Reformulation Prediction Methods

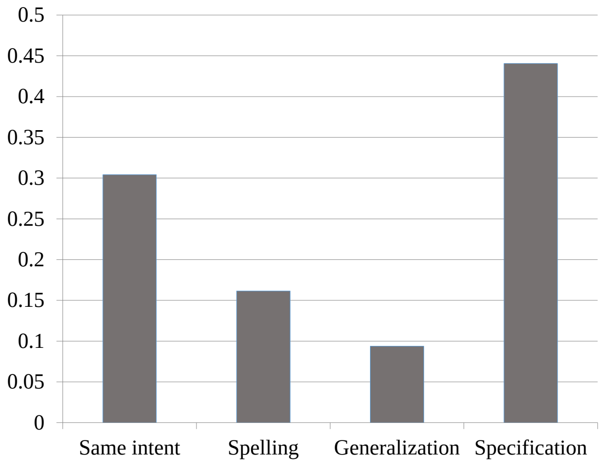

Figure 2 Distribution of Query Reformulation Types

5.2 Predicting Query Reformulation

In this section we describe the experiments we conducted to evaluate the reformulation prediction classifier. We perform experiments using the data described in the previous section. We compare the performance of four different systems:

- The first one, *Heuristic*, simply computes the similarity between two queries as the percentage of common words to the length of the longer query in terms of the number of words.

Table 2 : Heuristics vs. Textual vs. Concept Features for Reformulation Prediction

	Accuracy	Reform. F1	No-Reform. F1
Heuristics	77.10%	75.60%	69.07%
Textual	82.90%	71.75%	87.75%
Concepts	87.60%	81.20%	90.78%
All	89.02%	83.63%	91.75%

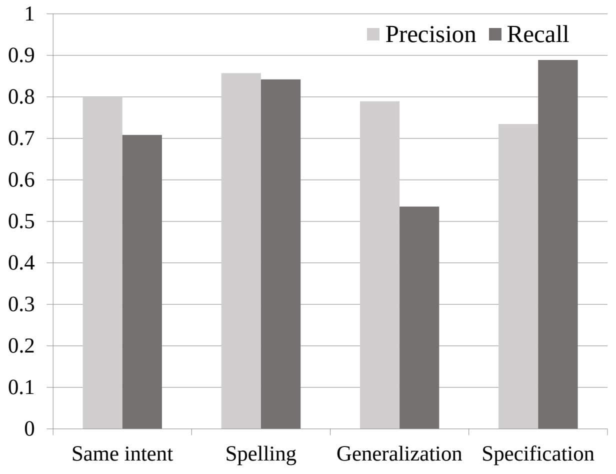

Figure 3 Precision and Recall for Query Reformulation Type Prediction

When finding common words, it allows two words to be matched if their Levenshtein edit distance is less than or equals 2. The second query is predicted to be a reformulation of the first if similarity $\geq \tau_{sim}$ and the time difference $\leq \tau_{time}$ minutes. The two thresholds were set to 0.35 and 5 minutes respectively using grid search to maximize accuracy over the training data.

- The second system, *Textual*, uses the textual features from previous work that have been described in Section 4.4.1 and the temporal and results features described in Section 4.4.3.

- The third system, *Concepts*, uses the concept features that we presented in Section 4.4.2 and the temporal and results features described in Section 4.4.3.

- Finally, the last system, *All*, uses both the textual features, the conceptual features and the temporal and results features.

For all methods, we used gradient boosted regression trees as a classifier with 10-fold cross validation. We also tried other classifiers like SVM and logistic regression but we got the best performance using the gradient boosted regression trees. All reported differences are statistically significant at the 0.05 level according to a two-tailed student t-test.

The accuracy, positive (reformulation) F1, and negative (non-reformulation) F1 for the four methods are shown in Table 2. The precision recall curves for all methods are shown in Figure 1; the heuristic method uses fixed thresholds resulting in a single operating point. The results show that the concept features outperform the textual features. Combining them together results in a small gain over using the concept features only. The concept features were able to achieve higher precision rates while not sacrificing recall because they were more effective in eliminating false reformulation cases.

We examined the cases where the classifier failed to predict the correct label to understand when the classifier fails to work properly. We identified several cases where this happens. For example, the classifier failed to match some terms that have the same semantic meaning. Many of these cases were acronyms (e.g. “AR” and “Accelerated Reader”, ”GE” and ”General Electric”). These cases can be handled by using a semantic matching method that yields higher coverage especially in cases of acronyms.

The classifier also failed in cases where the keyword extractor and/or the POS tagger failed to correctly parse the queries (e.g. “last to know”was not recognized as a song name). These cases can be handled by identifying named entities as a preprocessing step and treating them accordingly when identifying keywords or assigning POS tags to keywords.

Another dominant class of cases where the classifier failed were cases where the dependency rules failed to correctly identify the head keyword in a query. In many such cases, the query was a non well-formed sequence of words (e.g. “dresses Christmas toddler”). This is the hardest class to handle. Since it is hard to correctly parse short text and it is even harder when the text it is not well-formed.

5.3 Predicting Reformulation Type

We conducted another experiment to evaluate the performance of the reformulation type classifier. We performed experiments using the data described earlier where judges were asked to select the type of reformulation for every reformulation query. The distribution of reformulations across types is shown in Figure 2. The figure shows that most popular reformulations types are those where users move to a more specific intent or express the same intent in a different way. Reformulations with spelling suggestions and query generalizations are less popular. We conducted a one-vs-all experiment using gradient boosted regression trees with 10-fold cross validation. The precision and recall of every type are shown in Figure 3. The micro-averaged and macro-averaged accuracy was 78.13% and 72.52% respectively.

6 Conclusions

Identifying query reformulations is an interesting and useful application in Information Retrieval. Reformulation identification is useful for automatic query refinements, task boundary identification and satisfaction prediction. Previous work on this problem has adopted a bag-of-words approach where lexical similarity and word overlap are the key features for identifying query reformulation. We proposed a method for identifying concepts in search queries and using them to identify query reformulations. The proposed method outperforms previous work because it can better represent the information intent underlying the query and hence can better assess query similarity. We showed that the proposed method significantly outperforms the other methods. We also showed that we can reliably predict the type of the reformulation with high accuracy.

References

Peter Anick. 2003. Learning noun phrase query segmentation. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 88–95.

M. Arlitt. 2000. Characterizing web user sessions. *ACM SIGMETRICS Performance Eval Review*, 28(2):50–63.

Ricardo Baeza-Yates, Carlos Hurtado, Marcelo Mendoza, and Georges Dupret. 2005. Modeling user search behavior. In *LA-WEB ’05: Proceedings of the Third Latin American Web Congress*, Washington, DC, USA. IEEE Computer Society.

A. L. Berger and J. Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of the 22th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 1999)*, pages 222–229.

S. Bergsma and I. Q. Wang. 2007. Learning noun phrase query segmentation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 816–826.

Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. 2008. The query-flow graph: model and applications. In *Proceeding of the 17th ACM conference on Information and knowledge management (CIKM 2008)*, pages 609–618.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Doug Downey, Susan Dumais, and Eric Horvitz. 2007. Models of searching and browsing: Languages, studies, and applications. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(6):862–871.

J. Gao, X. He, and J. Nie. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In *Proceeding of the ACM conference on Information and knowledge management (CIKM 2010)*, pages 1139–1148.

J. Gao, S. Xie, X. He, and A. Ali. 2012. Learning lexicon models from search logs for query expansion. In *Proceeding of the Conference on Emprical Methods for Natural Language Processing (EMNLP 2012)*.

J. Guo, G. Xu, H. Li, and X. Cheng. 2008. A unified and discriminative model for query refinement. In *Proceedings of the annual international ACM SIGIR conference on Research and development in information retrieval*, pages 379–386.

M. Hagen, M. Potthast, B. Stein, and C. Brautigam. 2010. The power of naiv query segmentation. In *Proceeding of the ACM Conference of the Special Interest*

Group on Information Retrieval (SIGIR 2010), pages 797–798,.

M. Hagen, M. Potthast, B. Stein, and C. Brautigam. 2011. Query segmentation revisited. In *Proceeding of the ACM World Wide Web Conference (WWW 2011)*, pages 97–106.

J. Huang, J. Gao, J. Miao, X. Li, K. Wang, and F. Behr. 2010. Exploring web scale language models for search query processing. In *Proceeding of the ACM World Wide Web Conference (WWW 2010)*, pages 451–460.

Bernard J. Jansen and Amanda Spink. 2006. How are we searching the world wide web?: a comparison of nine search engine transaction logs. *Inf. Process. Manage.*, 42:248–263, January.

B. J. Jansen, A. Spink, and J. Pedersen. 2005. A temporal comparison of altavista web searching. *Journal of the American Society for Information Science and Technology*, 56:559–570.

B. J. Jansen, M. Zhang, and A. Spink. 2007. Patterns and transitions of query reformulation during web searching. *International Journal of Web Information Systems*.

Rosie Jones and Kristina Klinkner. 2008. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proceedings of ACM 17th Conference on Information and Knowledge Management (CIKM 2008)*.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substititions. In *Proceedings of the Fifteenth International Conference on the World-Wide Web (WWW06)*, pages 387–396.

Tessa Lau and Eric Horvitz. 1999. Patterns of search: Analyzing and modeling web query refinement. In ACM Press, editor, *Proceedings of the Seventh International Conference on User Modeling*.

C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. 20011. Identifying task-based sessions in search engine query logs. In *Proceedings of ACM Conference on Web Search and Data Mining(WSDM 2011)*.

Q. Mei, D. Zhou, and K. Church. 2008. Query suggestion using hitting time. In *Proceeding of the 17th ACM conference on Information and knowledge management (CIKM 2008)*, pages 469–478.

M. Mitra, A. Singhal, and C. Buckley. 1998. Improving automatic query expansion. In *Proceedings of the 21th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–214.

G. V. Murray, J. Lin, and A. Chowdhury. 2006. Identification of user sessions with hierarchical agglomerative clustering. *ASIST*, 43(1):934–950.

Seda Ozmutlu. 2006. Automatic new topic identification using multiple linear regression. *Information Processing and Management*, 42(4):934–950.

Filip Radlinski and Thorsten Joachims. 2005. Query chains: learning to rank from implicit feedback. In Robert Grossman, Roberto Bayardo, and Kristin P. Bennett, editors, *KDD*, pages 239–248. ACM.

Jamie Teevan, Eytan Adar, Rosie Jones, and Michael Potts. 2006. History repeats itself: Repeat queries in yahoo's logs. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 703–704.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceeding of the Human Language Technologies Conference and the Annual Meeting of the North American Association of Computational Linguists (HLT-NAACL 2003)*, pages 252–259.

K. Wang, C. Thrasher, and B. Hsu. 2011. Web scale nlp: A case study on url word breaking. In *Proceeding of the ACM World Wide Web Conference (WWW 2011)*, pages 357–366.

Ryen W. White and Steven M. Drucker. 2007. Investigating behavioral variability in web search. In *Proceedings of the 16th international conference on World Wide Web*.

Ryen W. White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd international conference on World Wide Web*, WWW '13, pages 1411–1420.

X. Yu and H. Shi. 2009. Query segmentation using conditional random fields. In *Proceedings of the Workshop on Keyword Search on Structured Data (KEYS)*, pages 21–26.