Asking What No One Has Asked Before: Using Phrase Similarities To Generate Synthetic Web Search Queries

Marius Paşca
Google Inc.
1600 Amphitheatre Parkway
Mountain View, California
mars@google.com

ABSTRACT

This paper introduces a method for automatically inferring meaningful, not-yet-submitted queries. The inferred queries fill some of the knowledge gaps between documents, on one hand, and known (i.e., already-submitted) queries, on the other hand. Thus, the inferred queries expand query logs and increase their coverage. New candidate queries are over-generated from known queries via phrase similarity data, then filtered against the set of known queries. The accuracy of the generated queries is computed using open-domain questions from standard question answering evaluation sets. Over the ranked lists of questions inferred for each of the evaluation questions, the precision reaches 0.9 at rank 50. The set of inferred queries is more than twice as large as the set of input queries.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Algorithms, Experimentation

Keywords

Query generation, synthetic queries, query templates, distributional similarities, Web search queries, query logs

1. INTRODUCTION

Background: Queries submitted collectively by Web search users indirectly convey human knowledge. Given the query *"how to replace a clutch on a honda civic"*, car enthusiasts would easily recognize that a particular car has parts that sometimes need replacement. Similarly, music aficionados would quickly interpret a query *"lyrics of yesterday beatles"* to refer to the lyrics of a particular song performed by a particular band. But not all queries lend themselves to easy interpretation. Indeed, queries take many forms, varying from relatively infrequent grammatically-structured, precise natural-language queries, to frequent noisy, underspecified, keyword-based queries. Throughout this continuum, the desire for better understanding of search queries is pervasive [7, 5], given its benefits towards better search results.

Motivation: If knowledge is generally prominent, users will eventually ask about it, especially as the number of users and the breadth of the available knowledge increase. But at a given time, an ever-growing document collection like the Web may capture more knowledge than has been asked about by search queries. Generating all meaningful queries that fill the knowledge gaps between documents, on one hand, and known (i.e., already-submitted) queries, on the other hand, is a daunting task. Manually compiling a comprehensive set of questions about a document or passage requires deep understanding of the document. Fortunately, some of the knowledge gaps are easier to fill than others. If a clutch is a replacement part on a *honda civic* and on a *truck*, it might be a replacement part on similar items including a *motorcycle* or *toyota yaris*, even if the query *"how to replace a clutch on a toyota yaris"* has not been submitted yet. Equally, if there are lyrics of *yesterday* and *hey jude*, there may be lyrics of *lovely rita* and *here comes the sun*, even if *"lyrics of here comes the sun beatles"* is not in the input set of queries.

Since inferred queries are new requests for knowledge derived from known requests for knowledge, they are useful in knowledge acquisition and access. For example, the development and training of specialized modules, such as those answering natural-language questions, benefit from the availability of additional inferred questions. In information extraction, methods operating over query logs rather than document collections benefit from a larger number of input queries, which increases the coverage of the extracted data. In Web search interfaces, inferred queries enrich the sets of queries displayed as potential completions based on what Web search users have typed so far, reducing the time to search result.

Contributions: We introduce a method for generating open-ended, not-yet submitted queries. The method aggregates known queries into query templates (e.g., *"lyrics of $\star$ beatles"*) associated with known phrase fillers (e.g., $\star\rightarrow${*yesterday*, *hey jude*}). The known phrase fillers of each query template are then expanded into new candidate phrase fillers. Queries generated in previous work have particular form (i.e., natural-language questions) and often require extensive linguistic processing of documents [14, 4]. In contrast, our method automatically generates new queries based on query analysis alone, as opposed to individual document analysis. This has the potential advantages of scalability and robustness with arbitrary, inherently-noisy queries. Furthermore, the method produces open-ended queries whose form resembles that of known queries, whether keyword-based or expressed in natural language. When applied to a large set of anonymized search queries, the method generates more than twice as many new queries as there are input queries. The accuracy of the generated queries is computed over ranked lists of new phrase fillers generated for question templates corresponding to open-domain questions from standard question answering evaluation sets [18]. Precision over the evaluation set of question templates, for which new phrase fillers are generated, reaches 0.93 at rank 10 and 0.90 at rank 50.

2. GENERATING NEW QUERIES

Definitions: A query Q is a sequence of tokens Q=[q_1 q_2 .. q_i .. q_{N_Q}], submitted as a search query by Web users.

A query template T is a generalization of a set of queries that share a common prefix [q_1 q_2 .. q_{i-1}] and a common postfix [q_{j+1} .. q_{N_Q}]. The template is obtained by replacing the variable infix [q_i q_{i+1} .. q_j] from each query, with a generic template filler $\star$: T=[q_1 q_2 .. q_{i-1} $\star$ q_{j+1}.. q_{N_T}]. The template filler $\star$ is associated with the set of infixes, or phrase fillers, from the queries that contributed to it. Each of the contributing queries can be reconstructed from the query template.

A template signature T_S is a generalization of a set of query templates that, if their token sequences are reduced to token sets (i.e., ignoring the relative order of the tokens), become the same. Some tokens that belong to a fixed, global set of tokens deemed irrelevant may be discarded in the process. The template signature is associated with the query templates that contribute to it. Using these definitions, the processing stages of our method for generating new queries are described in the following.

Aggregation into Query Templates: Our method takes as input a set of Web search queries. The sequence of tokens available in each query is split into all combinations of triples of a prefix, non-empty infix and postfix. For example, *"lyrics of hey jude beatles"* is split into *lyrics*, *of hey jude*, *beatles*; *lyrics of*, *hey jude*, *beatles*; *lyrics of*, *hey*, *jude beatles*; etc. Resulting triples that share a common prefix and postfix are aggregated into a query template, where the input infixes are the known phrase fillers of the template:

$$\left.\begin{array}{l}\text{lyrics of come together beatles}\\ \text{lyrics of hey jude beatles}\\ \text{lyrics of yesterday beatles}\end{array}\right\}\text{“lyrics of }\star\text{ beatles”}$$

where the template filler $\star$ corresponds to the set of known phrase fillers, i.e., infixes from the input queries: {*yesterday*, *hey jude*, *come together*}. An input query may contribute to the creation of multiple query templates, via different infixes. For example, another template created from *"lyrics of yesterday beatles"* is *"lyrics of yesterday $\star$"*:

$$\left.\begin{array}{l}\text{lyrics of yesterday toni braxton}\\ \text{lyrics of yesterday leona lewis}\\ \text{lyrics of yesterday beatles}\end{array}\right\}\text{“lyrics of yesterday }\star\text{”}$$

Generation of Candidate Phrase Fillers: In order to generate new queries, we expand the set of known phrase fillers into additional candidate phrases that may fill the query template. The expansion of phrase fillers is similar to the task of instance set expansion [15, 20], with a few differences. First, the instance set (i.e., the known phrases) to be expanded is derived automatically from noisy search queries, rather than provided manually as a clean set of seed instances. This is a crucial difference, since the choice of the instance set to be expanded has been shown to greatly affect the outcome and quality of the expansion, even with "correct", manually selected seeds [15]. Second, the set of known phrases varies in size across templates, from very small to very large. Third, it is not necessarily semantically coherent since it can span disjoint, unrelated classes: it includes *first* and *exxon valdez*, for the template *"when was the $\star$ oil spill"*; and *fragile x syndrome* and *men and women*, for *"what is the life expectancy for $\star$"*. Fourth, the expansion needs to be performed at large scale, i.e., individually for all query templates created from the set of input queries.

As a prerequisite to generating candidate phrase fillers, distributionally similar phrases [11, 12, 15] and their scores are collected in advance. Let $DS(K_i)$ be the list of most distributionally similar phrases of a known phrase filler K_i from a query template T. Any phrase U from $DS(K_i)$ is considered as a candidate phrase filler U of the respective query template. The score of U relative to the entire set of known fillers $\{K_i\}_{i=1}^{N}$ (and therefore, relative to T) is:

$$Sim(U,T) = \frac{\sum_{i=1}^{N} DSscore(U,K_i)}{N} \qquad (1)$$

where $DSscore(U,K_i)$ is the distributional similarity score [10] between U and K_i. For each template T, its candidate phrase fillers U are ranked in decreasing order of their scores. Known phrase fillers of T are discarded from the resulting list of candidate phrase fillers of T.

Filtering of Candidate Phrase Fillers: Candidate phrase fillers generated via distributional similarities are similar to known phrases only statically. To also take the context of the query template into account, the candidates are filtered using only the input queries:

1) A canonical, keyword-based template signature T_S is created from each template T. The process discards tokens from T that either are stop words, or are marked by a part-of-speech tagger [2] as prepositions (*of*, *for*), determiners (*the*, *an*), auxiliary verbs (*have*, *were*), or typical initial words in questions (*who*, *how*, *where*). The tokens not discarded are stemmed and ordered lexicographically in T_S. The same template signature may be shared by a large number of query templates:

$$\left.\begin{array}{ll}\text{“lyrics of }\star\text{ beatles”} & \text{“lyrics for }\star\text{ by the beatles”}\\ \text{“lyrics the beatles }\star\text{”} & \text{“}\star\text{ by beatles lyrics”}\\ \text{“beatles lyrics }\star\text{”} & \text{“lyrics }\star\text{ by the beatles”}\end{array}\right\}\text{“}\star\text{ beatl lyric”}$$

2) The list of candidate phrases of T is filtered, by retaining only known phrases of some other templates T' that share the same template signature T_S as T. For example, the unfiltered list of candidate phrases for the template *"lyrics of $\star$ beatles"* contains, among other phrases: *somehting*, *earlier today*, *last friday*, *gather together*, *eleanor rigby*, *two weeks ago*, *lucy in the sky with diamonds*, *strawberry fields forever*, *something else*, *here comes the sun*, *lovely rita*. Out of these phrases, *somehting*, *earlier today*, *last friday*, *gather together*, *two weeks ago*, *something else* are not among the known phrase fillers of any of the templates with the same signature as *"lyrics of $\star$ beatles"*. Therefore, these phrases are discarded from the candidate phrase fillers of *"lyrics of $\star$ beatles"*. In comparison, the query templates *"lyrics for $\star$ by the beatles"* and *"beatles lyrics $\star$"* have, among their known phrase fillers, the candidate phrases *eleanor rigby* and *lucy in the sky with diamonds*; and *here comes the sun* and *lovely rita* respectively. Therefore, these phrases are retained as candidate fillers of *"lyrics of $\star$ beatles"* after filtering.

A side effect of the filtering of candidate phrases is to effectively intersect vocabularies of candidate phrase fillers generated from documents, with vocabularies of known phrase fillers from queries. Indeed, candidate phrase fillers, generated from documents via distributionally similarities, are required to appear as known phrase fillers in other query templates of the same template signature.

The relative ranking of candidate phrases from the list of inferred unfiltered phrase fillers (before filtering) is preserved in the list of inferred filtered phrase fillers (after filtering). Each filtered phrase filler inferred for a query template corresponds to a new query, generated by filling the phrase into the slot filler of the query template.

Question Template (Original Question)	{Set of Known Fillers}: [Ranked List of Inferred Fillers]
what is the boiling point of ⋆ (What is the boiling point of water?)	{ch3br, caesium, engine oil, orange juice, gas,..}: [toluene, benzene, formic acid, acetonitrile, meoh, phenol, heptane,..]
when was the ⋆ *oil spill* (When was the Exxon Valdez Oil spill?)	{first, last, valdez, exxon valdez}: [past, torrey canyon, amoco cadiz, sea empress, braer, argo merchant, ixtoc,..]
what is the longest ⋆ *in the human body* (What is the longest bone in the human body?)	{organ, nerve, neuron, bone, cell,..}: [tendon, axon, bones]
who was the first african american to win the ⋆ (Who was the first African American to win the Nobel Prize in literature?)	{academy award, super bowl, medal of honor,..}: [tony award, grammy, grammy award, heisman, emmy, pulitzer, nobel,..]
how many home runs did ⋆ *hit* (How many home runs did Babe Ruth hit?)	{joe dimaggio, henry aaron, hank aaron, josh gibson, sammy sosa,..}: [albert pujols, chipper jones, ken griffey jr,..]
what ⋆ *helps prevent osteoporosis* (What mineral helps prevent osteoporosis?)	{}: [ ]
how fast is ⋆ *absorbed* (How fast is alcohol absorbed?)	{birth control, sugar, thc, whey protein,..}: [aspirin, proteins]
what is the life expectancy for ⋆ (What is the life expectancy for crickets)	{lupus, fragile x syndrome, men and women,..}: [copd, epilepsy, sarcoidosis, schizophrenia, hemochromatosis, sle,..]
how old do you have to be to get married in ⋆ (How old do you have to be to get married in South Carolina?)	{france, connecticut, kentucky, kansas,..}: [united states, us,..]
where did ⋆ *grow up* (Where did Golda Meir grow up?)	{sarah palin, lamar odom, jimmy buffet}: [drew barrymore, dakota fanning, keyshia cole, beyonce knowles, fall out boy,..]

Table 1: Sample of phrase fillers present in (Known) or generated (Inferred) queries, for a selected subset of the evaluation set of 457 TREC question templates. The method infers phrase fillers for 246 out of the 475 question templates

3. EVALUATION

3.1 Experimental Setting

Data Sources: The experiments rely on a random sample of around 100 million fully-anonymized queries in English submitted by Web users to Google in 2010. Each query is accompanied by its frequency of occurrence in the query logs.

A phrase similarity repository is available, which is derived following [12, 15] from unstructured text available within a sample of around 200 million documents in English. The repository provides data for each of around 1 million phrases that occur as full-length queries in the input query logs. It contains ranked lists of the top 200 phrases computed to be the most distributionally similar, for each phrase. For example, the top distributionally similar phrases for *caesium* are *cesium*, *rubidium*, *strontium*, *barium*, *thallium*, *lanthanum* etc.

Evaluation Set of Question Templates: An evaluation set of question templates is compiled from the factoid evaluation questions of the TREC Question Answering track [19, 18] from 1999 through 2003. For this purpose, a random sample of 800 TREC questions is manually inspected. The inspection of a question may result in one of three possible outcomes. First, the question may be discarded, if the resulting question templates would be so general that generating additional new phrases to fill it would be trivial (too easy). This occurs for 158 of the 800 questions (e.g., for *"what is the location of lake champlain"* or *"what is a fuel cell"*). Alternatively, the question may be discarded, if the question is too specific to generate any new queries from its possible templates. This is the case for 139 of the 800 questions (e.g., for *"what cuban dictator did fidel castro force out of power in 1958"*, *"what did john hinckley do to impress jodie foster"* or *"in what year did joe dimaggio compile his 56-game hitting streak"*). Finally, the question may be retained and manually converted into a question template, by changing a phrase from the question into a slot filler. A total of 503 of the 800 questions are thus converted into 457 unique question templates. The left part of Table 1 shows a sample of the evaluation set of question templates, along with the questions from which they were derived.

Extraction Parameters: Any query template is created from at least two distinct input queries, such that the slot of a template is filled by at least 2 known phrases. To cap computational costs, at most 10,000 candidate phrases are retained per query template, out of the phrases generated from the set of known phrases based on the phrase similarity repository. For each known phrase, the 200 most similar phrases available in the phrase similarity repository are considered. During the mapping of query templates into canonical template signatures for the purpose of filtering candidate phrases, all tokens are stemmed using the Porter stemmer [16].

3.2 Quantitative Results

Relative Coverage for Inferred Queries: For query lengths from 1 to 10 and also for all query lengths, the graphs in Figure 1 compute percentages of counts of unique queries, relative to counts of unique, original queries available in the input query logs. The left graph is for the original queries. It captures the distribution of unique queries over query length in the input query logs. In particular, it shows that most of the input queries contain between 2 and 4 tokens: 21.6% (2 tokens), 31.7% (3 tokens), 23.7% (4 tokens), with an overall 100% (all tokens). The middle graph is for the subset of original queries that contribute to the creation of any query template and one of its known fillers. Because no query templates are created from queries containing a single token, the percentage for query length 1 is 0. The middle graph shows that many of the original queries do contribute to the creation of query templates: 15.6% (2 tokens), 25.9% (3 tokens), 19.2% (4 tokens) of all unique original queries. Overall, 76.7% (all tokens) of the original queries contribute to creating query templates, which in turn generate new queries. Since query frequency has a long tail distribution, the three quarters of the original queries covered by our query generation method include frequent queries as well as many rare queries.

The right graph in Figure 1 is for inferred filtered queries, which do not include any of the original queries among them. Even after filtering, the number of inferred queries exceeds the number of original queries at all query lengths, with the exception of query length 2: 16.3% (2 tokens), 56.3% (3 tokens), 70.1% (4 tokens), with an overall 224% (all tokens). Thus, the addition of the set of inferred queries to the set of original queries more than triples the size of the set of original queries. The relative query-count increase, from original queries (left graph) to inferred queries (right graph), is higher important for relatively longer queries: 2.4% to 9.1% (7 tokens), 1.2% to 3.4% (8 tokens), 0.6% to 1.2% (9 tokens).

As a more precision-oriented alternative to the right graph in Figure 1, a separate experiment generates new queries from only the top 50 (as opposed to all) inferred filtered phrase fillers per

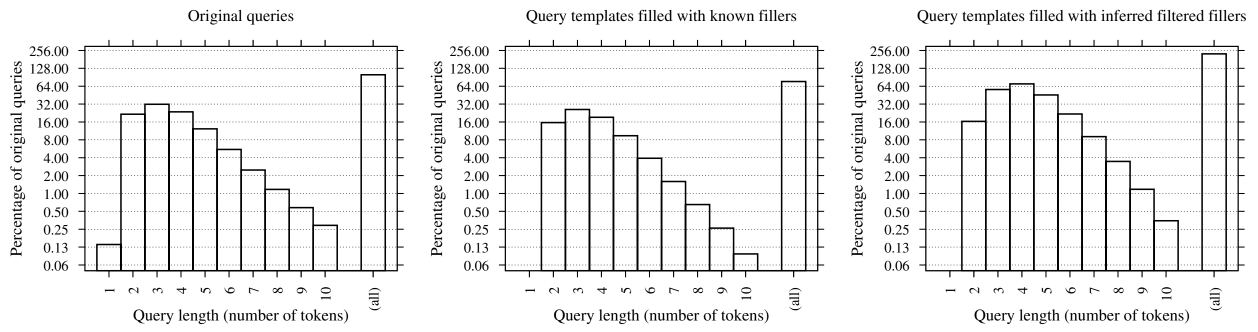

Figure 1: Number of unique queries of various lengths, shown as a percentage of the total number of unique original queries. Computed for the original query set (left graph), the query set obtained by filling each known phrase filler in its query template (middle graph), and the query set obtained by filling each inferred filtered candidate filler in its query template (right graph)

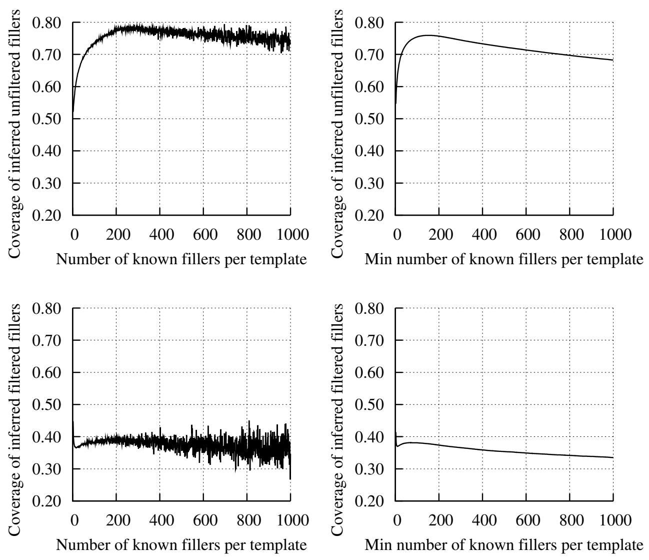

Figure 2: Coverage relative to known phrase fillers, for inferred unfiltered (upper graphs) or inferred filtered (lower graphs) phrase fillers, over templates containing a particular number (left graphs) or at least a particular number (right graphs) of known phrase fillers. Computed from the top 50000 inferred phrase fillers per template. For this computation only, known phrase fillers are not removed from the lists of inferred phrase fillers

query template. With this setting, the addition of the set of inferred queries almost doubles the size of the set of original queries, with 96% more queries.

Relative Coverage for Inferred Phrase Fillers: Because it is unfeasible to manually compile the exhaustive sets of phrases that can fill arbitrary query templates, we compute relative instead of absolute coverage. Figure 2 illustrates the coverage over all query templates, computed as the percentage of known fillers that occur among inferred fillers. Only for this computation, known fillers are not discarded from the ranked lists of at most 10,000 candidate fillers generated per query template. This is roughly equivalent to assessing to what extent candidate fillers generated from known fillers can recover known fillers that were to be missing. The left graphs of the figure show the relative coverage over query templates, when the absolute number of known phrase fillers per template increases. The right graphs correspond to a smoother version, in which the minimum number of known phrase fillers per template increases. The upper and lower graphs in the figure correspond to the coverage for inferred fillers before filtering and after filtering respectively. Naturally, filtering reduces relative coverage of known fillers, as shown by comparing the upper graphs against the respective lower graphs. Some of the known fillers (e.g., *men and women*) are not semantically similar to, and therefore may not be recovered from the similar phrases of, the other known fillers (e.g., *lupus*) of the template (e.g., *"what is the life expectancy for ⋆"*). This particularly affects query templates with few known fillers. Therefore, relative coverage in the upper graphs of Figure 2 is initially lower, but increases quickly as the (minimum) number of known phrase fillers for the template increases. For all graphs, coverage decreases slowly as the number of known fillers per template increases. The decrease is partly artificial, since an increase in the number of known fillers will cause more of the recovered known fillers to be ranked beyond the top 10,000 candidate fillers retained per query template.

Considering not all templates but just the evaluation set of 457 query templates, new queries are inferred (after filtering) for 246 of 457, i.e., 53% of the evaluation set. The average ratio of inferred phrase fillers vs. known phrase fillers is 290% or 159%, when computed over the subset of 246 or all 457 query templates respectively.

Forward Coverage for Inferred Queries: An additional experiment investigates the coverage of inferred queries, relative to queries submitted only later ("forward") in time. Concretely, forward coverage of inferred queries is measured relative to a separate "forward" sample of queries, used only for the purpose of this particular experiment. The forward sample consists in 100 million fully-anonymized queries in English, submitted collectively by Web users around 6 months later than the "main" query set (i.e., from which inferred queries are generated). Out of the forward query set, 55.6% of queries appear in the main query set. Conversely, almost half (44.4%) of the forward queries are new queries. More importantly, 8.6% of the queries in the forward query set are among the queries inferred from the main query set submitted 6 months earlier. In other words, 19.3% of the new queries in the forward query set can be inferred in advance from the main query set. Note that these queries represent only 3.5% of the set of inferred queries. Therefore, inferred queries capture queries submitted later in time, and also capture many other not-yet-submitted queries. If only the top 50 queries inferred per template are considered, the forward coverage numbers become 5.7% (instead of 8.6%) of forward

Label	Examples of *Question Template*: Phrase Filler
wrong	*what is the life expectancy for* ⋆: chinese crested
	when was the ⋆ *oil spill*: chernobyl nuclear
misspelled	*when did* ⋆ *enter the union*: illionis
	when was the ⋆ *oil spill*: exon valdez
okay	*what is the longest* ⋆ *in the human body*: bones
	how fast is ⋆ *absorbed*: proteins
correct	*what is the longest* ⋆ *in the human body*: tendon
	what is the boiling point of ⋆: toluene

Table 2: Correctness labels for the manual assessment of new phrases generated for query templates

Correctness	Percentage of Ranked Phrase Fillers							
	@5		@10		@20		@50	
	G_U	G_F	G_U	G_F	G_U	G_F	G_U	G_F
wrong	7.97	2.30	8.82	2.53	9.73	3.21	11.89	3.75
misspelled	5.28	1.43	5.24	1.68	5.16	1.68	5.08	1.63
okay	1.63	6.22	1.50	7.27	2.02	7.52	2.32	8.38
correct	85.12	90.05	84.44	88.52	83.09	87.59	80.71	86.24

Table 3: Comparative percentages of phrase fillers inferred for the evaluation question templates, whose correctness is deemed to be of a particular type from Table 2. Shown as an average over the subset of 246 question templates for which some phrase fillers were generated (G_U=inferred unfiltered candidate fillers; G_F=inferred filtered candidate fillers)

queries, and 12.8% (instead of 19.3%) of forward queries that are new queries.

3.3 Quality of Inferred Queries

Evaluation Procedure: The evaluation focuses on the assessment of accuracy of the ranked list of new phrases generated for each query template. Each phrase is manually assigned a correctness label within its respective query template, as shown in Table 2. A phrase is *correct*, if it is a meaningful filler of the template; *okay*, if it refers to a concept that would be a meaningful filler but as it stands grammatically disagrees with the template; *misspelled*, if it is a meaningful filler but is misspelled; or *wrong*, if it is not a meaningful filler.

Accuracy: Two experimental runs are evaluated, where new queries are generated by filling in a generated candidate phrase into its query template. In run $\mathbf{G}_U$, the ranked lists of generated candidate fillers are **u**nfiltered. Comparatively, in run $\mathbf{G}_F$, the ranked lists of candidate fillers are automatically **f**iltered.

The right part of the earlier Table 1 illustrates the ranked lists of filtered phrase fillers generated in run G_F, for a sample of the evaluation question templates. For example, the template *"what is the boiling point of* ⋆*"* has *caesium*, *orange juice* and *gas* among its known phrase fillers. The top fillers generated for it are *toluene* and *benzene*, which correspond to inferring the not-yet-submitted queries *"what is the boiling point of toluene"* and *"what is the boiling point of benzene"* respectively. No known fillers or inferred fillers are available for evaluation question templates such as *"what* ⋆ *helps prevent osteoporosis"*.

Table 3 illustrates how many of the phrase fillers are marked with each of the correctness labels from Table 2, at various ranks in the ranked lists of phrase fillers. Table 3 shows how, relative to G_U, run G_F reduces the counts of phrase fillers deemed as *wrong* and *misspelled*, but in doing so increases the number of phrase fillers deemed *okay*. More importantly, larger percentages of phrase fillers are deemed as *correct* for G_F than for G_U, across all ranks, e.g., around 90% vs. 85% at rank 5.

Inferred Query	Via Query Template
audio quality of ipod touch 3g	⋆ of ipod touch 3g
dress code at albert hall	dress code at ⋆
fly from paris to zagreb	fly from paris to ⋆
formatting decimal to string in c	⋆ decimal to string in c
how is the weather in key west in august	how is the weather in ⋆ in august
how to replace battery on imac	how to replace battery on ⋆
max speed of corvette zr1	⋆ of corvette zr1
pet supply stores in milwaukee	⋆ stores in milwaukee
what is the national flag of france	what is the ⋆ of france
when did reese witherspoon win the oscar	when did ⋆ win the oscar
where to buy t mobile phones without contract	where to buy ⋆ phones without contract

Table 4: Examples of queries inferred, after filtering (run $\mathbf{G}_F$), from the input set of known queries

Table 4 shows a small sample of inferred filtered queries, derived via arbitrary query templates (i.e., templates that are not restricted to the evaluation set of question templates). Existing methods that extract instance attributes from queries can derive additional attributes from inferred queries (e.g., *audio quality* for *ipod touch 3g* from the inferred query *"audio quality of ipod touch 3g"*, or *max speed* for *corvette zr1* from the inferred query *"max speed of corvette zr1"*), which may not be otherwise derived from the set of known queries.

Discussion: Although unfiltered candidate fillers have good quality for many query templates, their filtering is useful. Meaningful fillers of a query template should be similar to the known fillers not only statically, but also in the context of the query template. Unfiltered candidate fillers satisfy only the first requirement: their computation relies only on the set of known fillers, and ignores the context introduced by the remainder of the query template. Intuitively, unfiltered candidates will have lower quality, whenever the context introduces additional constraints that are more difficult to derive by analyzing solely the set of known fillers. For example, both templates *"what year was* ⋆ *born"* and *"what dress size was* ⋆*"* take person names as fillers. However, the presence of *dress size* in the latter restricts its acceptable fillers mostly to feminine names - an important constraint that may be difficult to capture automatically just from the known fillers. Indeed, the top 10 unfiltered candidate phrases generated for the latter template include *cary grant* and *james dean*, which do not correspond to meaningful generated queries. The phenomenon is even more visible for templates like *"when was* ⋆ *'s founded"* and *"who directed the film* ⋆*"*. Both templates have a moderate number of known fillers, many of which happen to look like first names of people: *wendy*, *tiffany*, for the former; and *jack*, *anand*, for the latter. As a result, the unfiltered generated phrase fillers are skewed towards people first names, most of which are irrelevant for the respective templates. Although imperfect, filtering takes advantage of constraints imposed on the candidate phrases by the presence of the same keywords in other queries, to discard irrelevant candidate phrases.

4. RELATED WORK

Our method infers new queries by exploiting a particular type of relatedness between sets of known queries, on one hand; and new queries generated from the query template corresponding to those known queries, on the other hand. In this light, an inferred query is related to a known query, in that the former can be inferred from the latter by replacing a phrase with another phrase from the same semantic class, as approximated via phrase similarities. Unlike previous work in the related area of computing related queries for a given query, our method does not require the availability of user feedback, in the form of query sessions or user-specific information [13, 3] or clicks on search results [3]. It does not require large amounts of click information or other labeled data, and does not bias the generated queries towards typical substitutions made by Web searchers to their queries [8]. Moreover, it attempts to generate new queries that are meaningful by themselves, in addition to being useful in some internal system module or task. Queries are also aggregated into query templates in [17], in order to compute related queries for a given query. Our mapping of multiple query templates into the same template signature is a simpler alternative to previous work on bridging longer questions with their equivalent shorter queries [9, 6].

Other previous work shares the intuition that submitted search queries are influenced by and indicative of various semantic relations holding among full-length queries or query tokens. Semantic relations are untyped, similarity-based relations from query logs in [1], and hold among full-length queries. Untyped relations can also be identified among query tokens, for the purpose of query reformulation [21]. Among the classes of query substitutions defined in [8], our generation of new queries from known queries fits under the class of approximate rewriting, i.e., substitutions that replace part of the query with an alternative from the same category.

5. CONCLUSION

In this paper, redundancy within Web search queries allows for the generation of not-yet-submitted queries that correspond to meaningful user information needs. Redundancy is used in two ways: through different queries that specify the same properties or same relations of similar phrases; and through queries that have different degrees of verbosity (from natural-language to keyword-based) but essentially use the same keywords. The analysis of query logs is limited to queries considered in isolation from one another; no search-result clicks or query sessions are needed.

Because queries inferred in this paper resemble existing queries, and existing queries tend to be limited in length, arbitrarily complex questions are unlikely to occur among inferred queries. Ongoing work explores the generation of new queries via double-slot (*"what did ⋆ do to impress ⋆"*), as opposed to single-slot, query templates; the identification of new queries that should be discarded because the inferred phrase filler does not agree syntactically with the remainder of the query.

Acknowledgments

We would like to thank Lev Finkelstein, for suggestions and technical support; and Yangbo Zhu and Leonid Velikovich, for access to distributional similarity data.

6. REFERENCES

[1] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *Proceedings of the 13th ACM Conference on Knowledge Discovery and Data Mining (KDD-07)*, pages 76–85, San Jose, California, 2007.

[2] T. Brants. TnT - a statistical part of speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP-00)*, pages 224–231, Seattle, Washington, 2000.

[3] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM Conference on Knowledge Discovery and Data Mining (KDD-08)*, pages 875–883, Las Vegas, Nevada, 2008.

[4] M. Heilman and N. Smith. Good question! Statistical ranking for question generation. In *Proceedings of the 2010 Conference of the North American Association for Computational Linguistics (NAACL-HLT-10)*, pages 609–617, Los Angeles, California, 2010.

[5] J. Hu, G. Wang, F. Lochovsky, J. Sun, and Z. Chen. Understanding user's query intent with Wikipedia. In *Proceedings of the 18th World Wide Web Conference (WWW-09)*, pages 471–480, Madrid, Spain, 2009.

[6] S. Huston and B. Croft. Evaluating verbose query processing techniques. In *Proceedings of the 33rd International Conference on Research and Development in Information Retrieval (SIGIR-10)*, pages 291–298, Geneva, Switzerland, 2010.

[7] B. Jansen, D. Booth, and A. Spink. Determining the informational, navigational, and transactional intent of Web queries. *Information Processing and Management*, 44(3):1251–1266, 2008.

[8] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of the 15h World Wide Web Conference (WWW-06)*, pages 387–396, Edinburgh, Scotland, 2006.

[9] J. Lee, S. Kim, Y. Song, and H. Rim. Bridging lexical gaps between queries and questions on large online qa collections with compact translation models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pages 410–418, Honolulu, Hawaii, 2008.

[10] L. Lee. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association of Computational Linguistics (ACL-99)*, pages 25–32, College Park, Maryland, 1999.

[11] D. Lin and P. Pantel. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational linguistics (COLING-02)*, pages 1–7, Taipei, Taiwan, 2002.

[12] D. Lin and X. Wu. Phrase clustering for discriminative learning. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP-09)*, pages 1030–1038, Singapore, 2009.

[13] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *Proceedings of the 17th International Conference on Information and Knowledge Management (CIKM-08)*, pages 469–477, Napa Valley, California, 2008.

[14] R. Mitkov and L. Ha. A computer-aided environment for generating multiple-choice test items. *Natural Language Engineering*, 12(2):177–194, 2006.

[15] P. Pantel, E. Crestan, A. Borkovsky, A. Popescu, and V. Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 938–947, Singapore, 2009.

[16] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[17] I. Szpektor, A. Gionis, and Y. Maarek. Improving recommendation for long-tail queries via templates. In *Proceedings of the 20th World Wide Web Conference (WWW-11)*, pages 47–56, Hyderabad, India, 2011.

[18] E. Voorhees. Overview of the trec 2003 question answering track. In *Proceedings of the 12th Text Retrieval Conference (TREC-2003)*, pages 54–68, Gaithersburg, Maryland, 2003.

[19] E. Voorhees and D. Tice. Building a question-answering test collection. In *Proceedings of the 23rd International Conference on Research and Development in Information Retrieval (SIGIR-00)*, pages 200–207, Athens, Greece, 2000.

[20] R. Wang and W. Cohen. Automatic set instance extraction using the Web. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP-09)*, pages 441–449, Singapore, 2009.

[21] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of the 17th International Conference on Information and Knowledge Management (CIKM-08)*, pages 479–488, Napa Valley, California, 2008.