An Automatic Query Augmentation Model for Answering Well-Defined Questions in Top-1 Search Result

Jing-Shin Chang
Department of Computer Science & Information Engineering, National Chi Nan University
1 University Road, Puli, Nantou 545, Taiwan, ROC.
jshin@csie.ncnu.edu.tw

Shu-Fan Shih
Department of Computer Science & Information Engineering, National Chi Nan University
1 University Road, Puli, Nantou 545, Taiwan, ROC.
s94321534@ncnu.edu.tw

Abstract

Short query terms often result in irrelevant search results due to the lack of appropriate contexts to disambiguate real user intention and thus introduce search errors. Without high precision raw search results, post re-ranking modules may not really help since garbage input only results in garbage output. Automatic query formulation, which supplies appropriate left and right contexts to the query terms, is therefore an important pre-processing technique for acquiring highly relevant documents and submitting them for post re-ranking.

A systematic approach for augmenting short query terms with the best contextual text patterns is proposed in this paper for matching answers of some well-defined questions such as "the birthday of Bill Gates" (and most factoid questions). The augmentation patterns are learned to directly maximize the top-1 accuracy rate for searching relevant documents. In comparison with the basic two-term query form, which submits a key entity query term ('Bill Gates') plus an intended attribute ('birthday') to be answered, the augmented patterns achieve 31% top-1 accuracy rate, in contrast to the extremely low, 4%, accuracy achieved by two-term query; the top-10 performance, which is about 54%, is also significantly better than the 21% accuracy with two-term query. This also implies that about 57% of the top-10 results have their correct answer given at the *first* place. By using appropriate augmented query terms, the correct search results can thus often be ranked at the first few places, and very likely at rank-1. Experiments show that the augmented query patterns significantly boost the top-1 performance for answering well-defined questions. By applying such techniques to queries, it is likely to improve the search precision significantly, sometimes even without the help of post re-ranking.

Keywords: query formation, query augmentation, augmented query term, user intention, search engines, well-defined questions, question answering.

1 Motivation

1.1 Search Errors with Back-End Ranking

Search engine is an important tool for finding answers to various questions from relevant documents. Since the huge web is also regarded as an important Web Corpus for developing academically interested language models, it is also now an important web navigating tool for searching linguistic patterns from the web. However, without high precision search results, the potential of all such applications will be limited.

Search engines had been improved significantly for the past tens of years by providing better front-end interfaces and back-end re-ranking models. From primitive AND-OR Boolean query operators to

Proceedings of the 22nd Conference on Computational Linguistics and Speech Processing (ROCLING 2010), Pages 17-29, Puli, Nantou,Taiwan, September 2010.

state-of-the-art page-ranking algorithms (Brin and Page, 1998), re-classification and re-clustering approaches (Zeng et al., 2004) and language models for relevance judgment (Gao et al., 2004), all efforts had been directed toward reducing the search space and providing high precision search result.

To reduce the search space and user burden, past research efforts had tried very hard to conduct some post-filtering steps based on various filtering criteria and re-ranking algorithms. For instance, some search engines, like Google, provide advanced search options and operators, such as "define:", "site:", "filetype:", "allinurl:", "allintitle:," to explicitly exclude raw search results that do not satisfy the specific semantics, regions of websites, file types or locations associated with the query terms or returned documents, in addition to traditional AND-OR operators. These kinds of operators explicitly reject many documents that might not fit user goal; however, they can in no way indicate which of the remaining documents match the user goal, like birthday of somebody. The users therefore still need to filter out their answers from many returned documents if they have no clever way to formulate their queries precisely.

State-of-the-art techniques had also been developed to identify the implicit user intention behind the query terms in order to reduce the search space. For instance, natural language query (NLQ) interfaces had been supported in the front end of the search engine and the key terms in the queries are expanded or transformed to search more relevant documents that might include desired answers (Wang and Wu 2003; Wu 1994). Since NLQ's normally embed keywords like "what", "when", "where", "who", "which", and "how", which are related to the question types of the queries, user intention can be more easily detected in such contexts. However, more documents than necessary might also be returned by inappropriate expansions or ambiguous transformations, which might results in some noisy documents. The result is an increase in recall at the cost of lower precision in the raw search results. As such, the user burden may not really be reduced.

Some of the spurious results might be due to the ambiguity of the query terms and their expansions. Re-clustering or re-classification algorithms (Zeng et al., 2004) are therefore applied to partition the returned documents by different senses of the query terms. Language models might also be applied to judge the relevance between the documents and the query terms (Gao et al., 2004). All these techniques help to reduce the search efforts of the user (if the expansion or transformation do not bring significant amount of noisy documents in the raw results.)

Even with so many prior efforts in back-end re-ranking, the most preferred search result may still not be ranked at the first few places. There are many reasons why a search engine might not rank the most preferred results at the *first* place. First of all, most ranking algorithms for presenting the search results know very little about the user intention or user goal for a particular query; the ranking may be simply based on the structures of incoming links and out-going links (Brin and Page, 1998), which do not directly related to the user interests associated with the query terms.

Secondly, short and ambiguous terms are often used in user queries. As a result, even with a search engine that attempts to figure out the user intention, it's not easy to identify the real user goal from the short queries. Unfortunately, in the front-end, users tend to submit short queries since they don't really know how to formulate effective queries to match the unknown answers for their questions. For instance, in searching the answer for the birthday of Bill Gates, a user may simply submit the key entity term 'Bill Gates' plus the attribute term 'birthday' to be answered to a search engine. Such a "two-term (entity-attribute) query form" maybe valid for very important persons like Bill Gates since a mass number of documents related to Bill Gates may eventually contain sentences like 'the *birthday* of *Bill Gates* is...'. But the birthday of an ordinary person may only be introduced informally in all relevant documents as 'somebody *was born on* ...', which may not be effectively matched with the two-term query since the 'birthday' attribute is expressed with another term which depends on the syntactic forms of the answering sentences.

Such short or inappropriate query terms therefore often result in irrelevant search results, since user goal or user intention cannot be easily identified without appropriate contexts, and the answering sentences may not be expressed directly in the same terms as the submitted entity-attribute terms or in a syntactic pattern that a user may think of.

Finally, and most critical of all, even though the user intention is clearly identified or specified with specific query terms, like '`birthday: Bill Gates`' ('when was Bill Gates born') and '`birthplace: Bill Gates`' ('where was Bill Gates born') in the front-end, the search engine still do not know effectively how the query terms should be formulated to match against the *unknown* an-

swers (i.e., 'October 28, 1955' and 'Seattle, Washington') in the wide variety of sentences that contain the answers, since there is no explicit links between the surface forms of the query terms and sentences containing the answers. Users simply cannot post higher level search options, like "birthday: Bill Gates" and "height: Eiffel Tower", to indicate their intentions to most search engines and rely on the search engine to automatically formulate appropriate query strings regardless of how the answers may be expressed in relevant documents.

As a result, the most preferred result is not always ranked at the *first* place. And, normally a huge number of noisy search results will also be returned even for a very simple and well-defined question (such as most factoid questions) with very clear answer. Sometimes the preferred answer may be returned in the first document, but the answer might be located very far away from the query terms. All these unsatisfactory results impose heavy load to the users and the users have to filter out a significant portion of the search results by hands.

Post re-ranking modules may not really help since garbage input only results in garbage output if the most relevant documents do not even have the chance to be submitted to the re-ranking modules. Therefore, being able to rank the most preferred document at the first few places in the search results is very important, with or without post re-ranking or clustering. Automatic query augmentation, at least for those queries with clear user intentions, is therefore an important pre-processing technique for acquiring highly relevant documents from a search engine and submitting them for post re-ranking, if necessary.

1.2 Solutions with Front-End Query Formulation

The fundamental problem is that the underlying indexing mechanism of current search engines only provides a pattern matching mechanism between the *query terms* and the *document* containing the query terms, but not the direct matching mechanism between the *query terms* and the yet unknown *answers*. Users actually get the unknown answers ('Oct. 28, 1955') by matching against the entity term ('Bill Gates') and the possible *contexts* ('birthday', 'of', 'the') of the answers, but the lexical and syntactic patterns of the contexts may have many possible forms, which cannot be easily guessed. The problem gets worse if the query terms are highly ambiguous or a large number of terms are synonymous to the query terms. In this case, more discriminative contexts may be needed to disambiguate the real user intention and the sentences containing such terms. That's why the search engines often return a huge number of documents that contains the query terms but contains no answer at all, and the best match, if indirectly found, may not be ranked at the first place. Under such a fundamental limitation of the current search engines, formulating appropriate query terms that might possibly match the answers in the documents is therefore an important technique to improve the search precision of the search result, hopefully ranking the document with the most appropriate answer at the first place.

Unfortunately, in many prior works, even though the user intention might be detected, the user intention was simply used at the *back-end* to filter out, re-rank, re-cluster or re-classify a large collection of documents; most such post processing modules do not have control over the quality of the raw search results. Some of the raw search results might be highly noisy. If this is the case, garbage raw search results might only produce garbage re-ranked output even with known user intention and the state-of-the-art re-ranking modules.

Observing this, we should use the user intention as early as possible and pay more attention at the *front-end* query formulation stage to get a collection of high precision raw search results and avoid submitting garbage to post processing modules.

2 Query Augmentation

The ability to return high precision raw search results at the *front-end* is therefore an important factor for the re-ranking modules to fully demonstrate their potential. Such ability is obviously related to the capability to match the query terms against those unknown answers in the documents with the help of known user intention.

To resolve such an answer-searching and matching problem, one can try to learn the possible lexical and syntactic patterns of the sentences containing desired answers associated with the query terms

and intention. The syntactic patterns for the answering sentences are then used to match the documents, instead of using some explicit and unformulated query terms.

For instance, to find the birthday of Bill Gates, we may augment the key term "Bill Gates" with its context "was born on", which is likely to appear near the answer, and submit the single augmented query token "*Bill Gates was born on*" to the search engine, instead of submitting "Bill Gates" plus "birthday" by heuristics. The documents matching this augmented query pattern will then likely to include the correct answer "(*Bill Gates was born on) October 28, 1955*". The augmentation pattern "*was born on*" for the explicit and required query term "*Bill Gates*" plays the role to restrict the search space to the particular user goal of "birthday". Unlike other query expansion methods, which may introduce a larger and expanded search space while expanding the query term and thus may introduce excessive search errors, we are actually shrinking the search space by using more precise lexical and syntactical patterns to match the answers, and not query-expanding them.

Such an augmented query pattern, inspired by a sentence pattern that contains the answer near the augmented query term, will be more effective in matching the answer than using heuristically selected query terms like "*Bill Gates*" AND "*birthday*", which may be far away in the matched documents, and thus unrelated to the answer. A natural language query like '*When is the birthday of Bill Gates?*' suffers from the same inefficiency in matching sentences containing the answer.

The reason why augmented query terms match better is that sentences answering the birthday of a person normally take the syntactic pattern of "$person was born on $date" instead of "The birthday of $person is $date." The latter syntactic form will sometimes be found, but only when $person is a really important person such that we have some documents that introduce this important person in such a straight and formal way. Also, there is rarely a chance to have a sentence like '*When is the birthday of Bill Gates?* It is *October 28, 1955*.' in any relevant documents. Therefore, there is also little chance that the question sentence will match the answer directly. Likewise, if the user goal is the "birthplace" of Bill Gates, the augmented query pattern "*Bill Gates was born in*" is likely to return the correct answer "Seattle, Washington" in the first place of the search results. Note that the most effect and discriminative query terms for these two highly related queries might differ in the prepositions 'on' and 'in', not in 'birthday' or 'birthplace'. This might suggest that intuitive query formulation does not always be the most effect. A systematic approach is required to learn the augmentation patterns to make the search really effective.

If we can formulate appropriate augmented query terms, with the help of known user intention, and the augmented query terms effetely match part of the potential answering sentences, then the most preferred documents containing the answer to the query is likely to be ranked at the first few places in the search results. In the best case, the answering documents will hopefully be ranked at the *first* place, even without the necessity of other re-ranking modules at the back end of the search engine. We therefore propose, in this paper, a simple and effective model for acquiring the augmentation patterns that can match the answering sentences for some query terms with known user intention. The goal is to acquire accurate search results that return the most preferred documents and answers in the first few places by augmenting user queries with appropriate augmentation patterns.

With such a query augmentation model, it will be possible to provide search engines with advanced search options like "birthday: Bill Gates" and "height: Eiffel Tower", corresponding to the natural language query 'when is the birthday of Bill Gates' and 'how high is the Eiffel Tower', for many frequently asked and well-defined questions whose answers can be found within the same sentence as the augmented query terms.

Note that, in this setting, we don't really need the search engine to judge the user intention from a short query, which is known to be very hard even with the current state-of-the-art works. Instead, the list of frequently asked attributes (such as 'birthday' and 'height') about entities is simply provided for user selection by the search engine. The search engine simply augments the short query with appropriate contextual patterns learned for the selected attribute. If the learning and augmentation methods, such as that proposed in the current work and others, prove to be effective, it will be worth the cost to build the augmentation patterns inside the search engine from the mass number of query-answer logs by classifying them by their attributes, either automatically or semi-automatically. This might be more realistic than using immature identification techniques for user intentions at the current stage of the art.

The same technique might also be applicable to natural language applications which require linguistics patterns in the Web Corpus that satisfy some known attribute.

In the following sections, a model for systematic and effective training of query augmentation patterns is proposed to maximize the top-1 searching criteria. Experiments results are then shown to demonstrate how such augmentation improves the searching performance in comparison with unaugmented queries.

3 Learning Augmentation Patterns

3.1 Well-Defined Questions

It is not always easy to learn augmentation patterns for all types of general questions. However, many well-defined questions can be answered by augmenting key query terms with their contexts and submitting them to a search engine. To make the search more effective, we further impose the requirement that the answer must appear in the same sentence as the key query terms.

By well-defined questions (WDQ's), we mean questions that can be answered (almost) unambiguously with a few words. For instance, the birthday, birthplace, height, weight of a person, the time for some important events, the author of a book, the capital of a nation, the principal of a university, the constellation of a movie star, and so on, are questions of this category. These kinds of problems are found everywhere, covering all well-known question types of "what", "when", "where", "who", "which", and "how" questions. Table A-1 gives some examples of the well-defined questions. The last column in the table provides examples of syntactic patterns that contain the answers of the questions.

3.2 Generating and Ranking Augmented Query Patterns

Given a set of query-answer pairs, the best augmentation patterns, which combine the key query term and its contexts, and which directly maximize the top-1 accuracy rate, can be learned easily with the following algorithm:

1. Submit the query-answer pairs to a search engine. Take the returned snippets and titles into account to extract the augmentation patterns. This step tells us how the answering sentences are expressed in the real world documents.
2. Extract the left token, middle tokens and right token separated by the query-answer pairs in the returned snippets. In general, we will have 5 parts, "($left) ($query|$answer) ($middle) ($answer|$query) ($right)", in a returned snippet near the query-answer pair. The tokens can be identified by a tokenizer or a word segmentation module. Currently, only one left token and one right token are considered in the augmented query patterns.
3. Generate the augmented query patterns by removing the $left or $middle or $right parts of the above 5 parts. In other words, an augmented query pattern is a subsequence of the above 5 parts, excluding the $answer. This step thus generates 8 different augmented query patterns for each returned snippet. Each such pattern represents a possible way to match the possible syntactic form of some answering sentences.
4. Remove augmented patterns that occur less often than a reasonable threshold to ignore rare syntactic patterns. In the current work, we use a cut-off value of 5 times.
5. Re-submit the query-answer pairs with the 8 different types of augmented left, middle, or right contexts. Identify the rank of the first snippet whose query-answer pair is in the same sentence. Note that adjacent tokens are doubly-quoted when submitted to the search engine so the engine only returns snippets whose query and answer terms are close to each other or even in the same sentence.
6. Rank the augmentation patterns by their searching performance for all the query-answer pairs. We virtually used the Top-1 performance as the ranking criteria in this work.

Theoretically, we can use the top-1 performance to select the best augmented patterns. However, the search engine may not be able to return the best documents in the first place in the training phase due to its own imperfect ranking criteria for the some query-answer pairs. Therefore, we use 10*Top-1 + 1 * Top-10 as the ranking score for extracting the best augmentation patterns. Since Top-1 accuracy is

weighted much higher, the auxiliary criteria for Top-10 performance will not seriously affect the major Top-1 criteria in learning; it simply backs off to other high rank results when Top-1 result is not available.

The above training process is similar to (Ravichandran and Hovy 2002). It's not absolutely new, but it is proved to be the most effective for some TREC tasks. However, the major criterion used by (Ravichandran and Hovy 2002) is the percentage of times a contextual pattern is used to get the correct answer for a query; it does not care whether the correct answer is given at the first place or at the 100^{th} place. The proposed method here avoided this by directly using the top-1 performance as the major criterion, and thus is likely to acquire augmentation patterns, that would get the best answer at the *first* place most of the time.

With this simple training algorithm, we can virtually train the augmented patterns to directly maximize the Top-1 searching performance, instead of other criteria that are not directly related to the IR performance in other methods (Voorhees 2001; Ravichandran and Hovy 2002).

4 Experiments

4.1 Data Sources

To evaluate the proposed augmentation mechanism, we shall show some experiments on a set of 122 query-answer pairs in details, where the key query terms are names of artists (including singers, movie starts, and so on) and the intended answers are the horoscopes or constellations (星座) of the artists. These pairs are divided into training and test sets for learning augmentation patterns and evaluating the performance, respectively. A few examples of the pairs are given in Table A-2 in the Appendix. Other training pairs had also been used in a series of experiments, including authors of books, capitals of countries, principals of universities, directors of movies and singers of albums. We will show the results of these WDQ's later to demonstrate the effectiveness and interesting observations across different domains. Other than such cross-domain behavior, they have similar results and conclusions as the constellation problem.

4.2 Baseline Models

The evaluation is compared with two baselines, each simulating one of the most intuitive query forms for general users with simplest query formulation. The first baseline model (baseline-1, or B1) is called "single-term query", and the second baseline (baseline-2, or B2) is called "two-term query".

The single-term query simply submit the key entity term (in the current case, an artist's name) to the search engine (Google) and see if the search engine returns the constellation of the artist in the first document with the answer in the same sentence as the key query term. This baseline model can simulate the case where user intention is completely unknown.

The two-term query submits both the key entity query term (i.e., artist's name) and the representative attribute word (i.e., "星座" "constellation") that represents the answer to be found. Before these two terms are submitted, they are virtually ANDed (in a loose way). This baseline simulates the general case where the user intention is known and an attribute word for the key term is used but only the simplest method (AND operation) is used to formulate the query.

The augmented query, on the other hand, augments the key entity term with the most discriminative lexical form of the attribute as well as other discriminative contexts in an appropriate syntactic form.

4.3 Comparison with Augmented Queries

The comparison of the augmented queries with the baselines is demonstrated in Figure 1 and Table 1, in terms of various performance criteria.

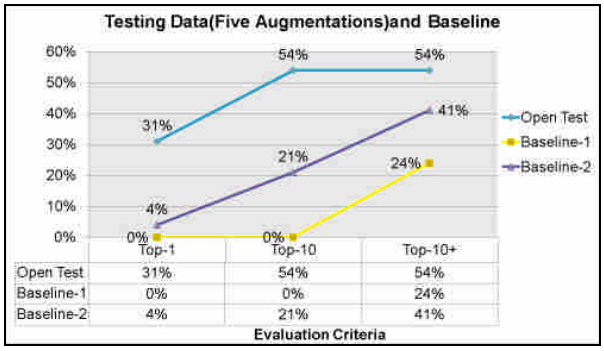

Figure 1. Test Set Performance for Augmented Query (Five Best Augmentations) and Baselines.

Model	Top1	Top10	Top10+	ARR	AR
B-1	0%	0%	24%	0.0010	1010.6
B-2	4%	21%	41%	0.0055	181.6
Open	31%	54%	54%	0.1031	9.7
Close	39%	54%	68%	0.1450	6.9

Table 1. Comparison of Augmented Query with Baselines.

The above table demonstrates the results for baseline-1 (B-1), baseline-2 (B-2) and augmented query using the Best-5 augmented query strings for the test set (Open) and training set (Close), respectively. The performance is evaluated in terms of the Top-1 accuracy rate, Top-10 including rate, average reciprocal rank (ARR) and average rank (AR). The percentage of queries whose correct answers can be found outside the 10th place is also listed as 'Top-10+' for reference.

Without any mechanism for forming good query strings, the first correct answer is rarely found in the first returned document. Submitting only the key query term never gets the right answer in the first document or in the first 10 documents. If the key query term is ANDed with a typical attribute word to restrict the search space, a better result may be possible. But the 4% Top-1 performance and the 21% Top-10 performance is still not impressive.

The situation is changed drastically when the key query term is augmented with its contextual patterns; the Top-1 performance is boosted significantly from 4% to 31% and the Top-10 performance is increased from 21% to 54%. On average, the answer can be found in the first 9.7 documents returned by the search engine. Note that the baselines are conducted with all query-answer pairs. Therefore, if we compare them with the combination of the training (close) and test (open) data, the difference will be even larger.

The improvement can be attributed to several factors. First of all, the best augmentation patterns will restrict the search space with different attribute words that are related the answer we are searching for. For instance, the augmented patterns may include not only 'birthday' but also *synonymous variants forms* like 'birth' or 'born', whichever is most effective, when the user is interested in the birthday of Bill Gates. Secondly, the augmented patterns will partially matches the most frequently used *syntactic patterns* (or an n-gram grammar) that could be used to write an answering sentence. For instance, to express the birthday of Bill Gates, a variety of sentence patterns can be used, and such patterns will be learned by the query augmentation mechanism. Therefore, the query augmentation mechanism provides a way to write the query with right attribute words and right syntax such that the document containing right answer can be matched at the surface level without resorting to a complicate re-ranking model at the back-end of the search engine.

4.4 Sensitivity to Training Data Size

It may be a problem if a large number of question-answer pairs are required to effectively learn the augmentation patterns. Fortunately, the words and the syntax which are used frequently to describe an attribute of an entity are not as wild as one may expect.

To investigate the sensitivity of the query augmentation mechanism to data sizes, we used different training data sizes (derived from the 122 pairs) to learn the augmentation patterns. We then estimate their training set performances, in terms of the Top-1, Top-10 and Top-10+ criteria, with the number of the best augmentation patterns as the only independent variable.

It is observed that the performance curves converge quickly when the best 5 or best 10 patterns are used to augment the key query terms. To avoid over fitting, using the best 5 patterns seems to be a good balance. By inspecting the Best-5 query augmentation patterns, as listed in Table 2, one can see that the 4 subsets of training data, which use 20%, 30%, 60% and 90% of the whole data set respectively, actually have the same best-5 augmentation patterns. The 5 best patterns are simply ranked differently with different training sizes.

Rank	QAug of 20% Data	Aug Type	QAug of 30% Data	Aug Type
1	"座的 Query"	AMQ	"座的 Query"	AMQ
2	"Query 是""座"	QMAR	"座的 Query 對"	AMQR
3	"座的 Query 對"	AMQR	"座 Query"	AMQ
4	"座 Query"	AMQ	"Query 是""座"	QMAR
5	"Query 星座"	QMA	"Query 星座"	QMA

Rank	QAug of 60% Data	Aug Type	QAug of 90% Data	Aug Type
1	"座的 Query"	AMQ	"座的 Query"	AMQ
2	"座 Query"	AMQ	"座 Query"	AMQ
3	"Query 是""座"	QMAR	"Query 是""座"	QMAR
4	"座的 Query 對"	AMQR	"Query 星座"	QMA
5	"Query 星座"	QMA	"座的 Query 對"	AMQR

Table 2. The Best-5 Query Augmentation Patterns and Augmentation Types Trained from Training Data of Different Sizes.

In the above table, the variable "Query" in the query augmentation (QAug) patterns refers to the key query term, which is a person name in the current experiments. The augmentation type (AugType) is encoded with the 5 letters, Q/Query, A/Answer, L/Left, R/Right, M/Middle, according to the constituents of an augmented query pattern. For instance, AMQ indicates that the answer (A) can be found from the left hand side of the augmented query string "MQ".

It can also be found in the above table that the best augmentation pattern (i.e., "座的 Query" in Rank-1) is consistently the best pattern among all training data sizes. This further enforces the conclusion that the syntactic patterns and attribute words associated with an answer are not necessarily too wild. Therefore, training the augmented patterns can be data-efficient.

The tables also suggest that, if one is interested in the constellation of the famous movie director named "李安" (Ang Lee), then she/he can simply submit the augmented query string "*座的*李安" as a single token to the Google search engine. It is then likely to get the answer "天秤*座*" "Libra" in the first returned document immediately to the left of the augmented query term "座的李安". This augmented query string is simply the concatenation of the suffix '座' of a constellation name and the Chinese possessive marker '的' 'de' plus the key query term. Although the augmented query string is not a normal word or compound, it turns out to be the most effective 'term' in finding the constellation of a person exactly from the first document returned by the search engine.

4.5 Cross-Domain Experiments

In addition to the constellation problem, 5 additional sets of Q-A training pairs had also been used in a series of experiments, including authors (作者) of books, capitals (首都) of countries, principals (校長) of universities, directors (導演) of films and singers (演唱者, 歌手) of albums. The results of these WDQ's are compared here to show the effectiveness of the augmentation techniques and some interesting observations across different domains.

Well-Define Questions	Question Sentence	Number of Training Data
作者	紅樓夢的作者是誰？	100
首都	中華民國的首都在哪？	100
校長	國立台灣大學的校長是誰？	100
導演	斷背山的導演是誰	100
演唱者	七里香專輯演唱者是誰？	70

Table 3. Five Additional Types of WDQ's and Statistics of Training Data.

Table 3 indicates the questions types of interests and the numbers of training pairs used for the cross-domain experiments. The following tables give some examples of the real question-answer pairs for each question types.

No	作者($Answer)	書名($Query)	首都($Answer)	國家($Query)
1	三毛	哭泣的駱駝	亞的斯亞貝巴	衣索比亞
2	李家同	讓高牆倒下吧	都柏林	愛爾蘭共和國
3	陳之藩	旅美小簡	塔林	愛沙尼亞
4	劉墉	我不是教你詐	安道爾城	安道爾
5	高陽	慈禧全傳	羅安達	安哥拉

Table 4. Training Examples for the Authors of Books and Capitals of Countries.

No	校長($Answer)	學校($Query)	導演($Answer)	電影($Query)
1	李嗣涔	國立台灣大學	陳凱歌	無極
2	陳文村	國立清華大學	于仁泰	霍元甲
3	吳重雨	國立交通大學	周星馳	長江七號
4	吳思華	國立政治大學	吳宇森	英雄本色
5	吳妍華	國立陽明大學	李安	斷背山

Table 5. Training Examples for the Principals of Universities and Directors of Movies.

No	演唱者($Answer)	專輯($Query)
1	周杰倫	七里香
2	王力宏	心中的日月
3	林俊傑	西界
4	潘安邦	美好時光
5	費玉清	浮聲舊夢

Table 6. Training Examples for Singers of Albums.

By acquiring the augmentation patterns from about 40 training pairs with the previously mentioned training process, the Top-5 augmentation patterns for the above 5 question types are shown as follows.

Rank	QAug. of Author	Aug. Type	QAug. of Country	Aug. Type
1	"Query 作者"	QMA	"Query 首都"	QMA
2	"寫的 Query"	AMQ	"位於 Query 首都"	LQMA
3	"Query""作品"	QAR	"在 Query 首都"	LQMA
4	"筆下的 Query"	AMQ	"從 Query 首都"	LQMA
5	"Query""著"	QAR	"抵達 Query 首都"	LQMA

Table 7. Best Augmentation Patterns for the Authors of Books and Capitals of Countries.

Rank	QAug. of School Principal	Aug. Type	QAug. of Film Director	Aug. Type
1	"由 Query 校長"	LQMA	"導的 Query"	AMQ
2	"與 Query 校長"	LQMA	"Query 導演"	QMA
3	"Query 新任校長"	QMA	"Query 導演""主演"	QMAR
4	"的 Query 校長"	LQMA	"和他的 Query"	AMQ
5	"和 Query 校長"	LQMA	"Query 導演""編劇"	QMAR

Table 8. Best Augmentation Patterns for the Principals of Universities and Directors of Films.

Rank	QAug. of Singer	Aug. Type
1	"Query 試聽下載"	QMA
2	"Query 專輯"	AQR
3	"Query 新歌"	AQR
4	"歌曲 Query"	LQA
5	"引""Query"	LAQ

Table 9. Best Augmentation Patterns for the Singers of Albums.

These tables show that we will effectively know that the author of 《哭泣的駱駝》 is "三毛" by posting a query like "哭泣的駱駝作者", instead of "請問哭泣的駱駝是誰寫的？" In asking for the capital of a country, such as 愛爾蘭共和國 (the Irish), we can post "愛爾蘭共和國首都" ("Irish capital") or add prepositions like "位於" ("at"), "在" ("in") , "從" ("from"), or some verb like "抵達" ("arrive at") before the country name "the Irish" and the keyword "capital" to find its capital immediately to the right of the augmented query string in the first few search results. For querying the principal of a university, we can also improve the search accuracy by pre-pending some prepositions to the name of the university and the keyword "principal", formatting a query like "由國立清華大學校長". On the other hand, Chinese query strings like "導的斷背山" ("directed by, Brokeback Mountain") and "七里香試聽下載" ("Common jasmin orange, download for listening") are likely to find you, respectively, the director of the "Brokeback Mountain", 李安 Ang Lee, and the major singer, 周杰倫 Jay Chou, for the album entitled "七里香" in the top few search results.

To see the effects of corpus size on the search performance across different question types, we further select two training sizes, which are about 40 pairs and 100 pairs, respectively for each question types and compare their Top-1 performance. Table 10 demonstrates the different performances and pattern sizes among the 6 question types. The '#Data' column indicates the number of training question-answer pairs; for each question type, a smaller training set (of about 40 pairs) and a larger training set (of about 100 pairs) are used. The #Permutation indicates the number of possible augmented query strings segmented from the left and/or right contexts of the main query keyword and its answer. Since this is quite a large number, two thresholds are applied to reduce the number of such augmented query strings. Threshold-1 simply truncates augmented query strings that occur less than 5 times. Threshold-2 further truncates augmented query strings that do not contribute to the Top-10 performance significantly. With these two truncated sets of augmented query strings, the best searching performance actually converges at the first few most effective syntactic patterns. The number of patterns

with which the searching performance no longer increases significantly is indicated in the 'Converge' column. And, the Top-1 performance with these limited number of augmentation patterns is shown in the 'T1%' column.

From this table, it is clear that the constellation problem is actually harder than all the others questions but it has simpler query forms. Most of the question types can be answered effectively using only tens of augmented query strings. Some question types, such as the singer problem, can actually be searched very effectively by applying the query augmentation techniques proposed in this paper. And, the number of training pairs is normally small. Therefore, it is possible to build, for many interesting questions, the augmented query string patterns with very little costs.

WDQ	# Data	# Permutations	Threadhold-1	Threadhold-2	Convergence	T1%
星座	39	3960	80	23	5	50%
星座	109	14676	379	31	5	43%
作者	41	19528	692	253	40	80%
作者	100	41630	1687	653	45	78%
首都	40	25657	881	110	15	85%
首都	100	47677	1803	212	20	89%
校長	40	11750	400	142	5	83%
校長	100	14223	463	147	25	88%
導演	40	23807	782	114	30	55%
導演	100	45970	1682	164	30	52%
演唱者	39	26296	1324	903	14	100%
演唱者	70	48328	2389	376	15	99%

Table 10. Performance with Small and Large Training Sizes for All the Question Types.

5 Concluding Remarks

Inappropriate query strings in the *front* end of the search engine will pass garbage inputs to its *back* end, which is responsible for re-ranking the raw search results of the internal indexing module. The result is often unsatisfactory. An automatic query augmentation model and its training method are proposed in this paper in the hope that the augmented query string can directly drive the search engine to return the expected answer in the *first* returned document and in the same sentence as the key query term without the need for a post-ranking model. In other words, we are seeking for an automatic query formulation mechanism that is trained to maximize the Top-1 accuracy, instead of other criteria. The result is impressive, and the "garbage-in garbage-out" situation for post-ranking is greatly reduced.

It is observed that 31% of the augmented queries achieve this goal; and, 54% of them extract the expected answers from the first 10 returned documents and in the same sentence where the key term appears. In contrast, using the key terms only never get the expected answers in the best 10 documents. Even with the attribute words ANDed with the key terms, only 4% and 21% of the queries get their answer in the Top-1 and Top-10 document lists, respectively. Such an augmentation mechanism is therefore effective in the retrieval of answers for simple attributes of many entities.

Furthermore, it is observed that only about 5 best augmented query patterns will be sufficient to reach the best performance. The reason is that more training data will have little effects on the first few best augmentation patterns; they simply have different ranks in the top-5 list. Hence, this augmentation mechanism can be trained with a small training set.

Appendix

Question Type	Question	Sentence with Answer
What	星座	天秤座的劉德華
	血型	O 型的王建民
Who	作者	曹雪芹所著的紅樓夢
	校長	暨南國際大學的校長是張進福
When	日期	12 月 26 日發生的南亞大海嘯
	生日	1980 年 3 月 31 日出生的王建民
Where	路名	南海路的建國中學
	首都	中華民國的首都是台北
How	高度	1,612 米高的武當山

Table A-1. Examples of Well Defined Questions.

	星座(Answer)	人物(Query)
1	牡羊	張信哲
2	牡羊	黃子佼
3	牡羊	柯以敏
4	牡羊	成龍
5	牡羊	梁詠琪
6	牡羊	彭佳慧
7	金牛	林憶蓮
8	金牛	李翊君
9	金牛	張宇
10	金牛	張震嶽
11	金牛	王力宏
12	金牛	趙詠華
13	金牛	鍾麗緹
14	金牛	周渝發
15	金牛	李敖

	星座(Answer)	人物(Query)
16	金牛	張大千
17	金牛	王貞治
18	處女	張清芳
19	處女	江蕙
20	處女	陳曉東
21	處女	陳慧琳
22	處女	蘇有朋
23	處女	蔡依林
24	處女	許如芸
25	處女	安室奈美惠
26	天秤	劉德華
27	天秤	鄭伊建
28	天秤	徐熙媛
29	天秤	金城武
30	天秤	林志穎

Table A-2. Examples of Query-Answer Pairs for Model Training (星座: constellation, 人物: person name, 牡羊: Aries, 金牛: Taurus, 處女: Virgo, 天秤: Libra).

References

[1] Baeza-Yates, R. A. and Ribeiro-Neto, B., 1999. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc, Santiago, Chile.

[2] Brin, S. and Page, L. 1998. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." *Proceedings of the 7th World-Wide Web Conference (WWW7).*

[3] Cutting, D. R., Karger D. R., and Pederson J. O. 1993. "Constant Interaction-Time Scatter/Gather Browsing of Very Large Document Collections." *In Proceedings of the 16th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, pages 125-135, Pittsburgh, PA, 1993.

[4] Gao, Jianfeng, Jian-Yun Nie, Guangyuan Wu and Guihong Cao. 2004. "Dependence Language Model for Information Retrieval", *In Proceedings of SIGIR-2004.* Sheffield, UK, July 25-29, 2004.

[5] Google search engine, 2010. http://www.google.com/.

[6] Han, J. and Y. Fu, 1999. "Mining Multiple-Level Association Rules in Large Databases." *IEEE Trans. Knowledge and Data Engineering*, vol. 11, no. 5, Sept. 1999.

[7] Harman, D. K. 1995. "Overview of the Fourth Text Retrieval Conference (TREC-4)." *In: TREC-4*, pp 1-24.

[8] Hearst, M. A., Pedersen J. O. 1996. "Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results." *In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, Zurich, June 1996.

[9] Heydon, Allan and Marc Najork, 1999. "Mercator: A Scalable, Extensible Web Crawler", *World Wide Web*, vol. 2, no. 4, pp. 219--229, 1999.

[10] Kawano, H. and T. Hasegawa, 1998. “Mondou: Interface With Text Data Mining for Web Search Engine.” *IEEE Proc. 31st Annual Hawaii Int. Conf. on System Sciences*, pp.275-283, 1998.

[11] Kleinberg, J. 1998. “Authoritative Sources in a Hyperlinked Environment.” In *Proc. Ninth Ann. ACM-SIAM Symp*. Discrete Algorithms, pages 668-677, ACM Press, New York, 1998.

[12] Kleinberg, J. 1999. “Authoritative Sources in a Hyperlinked Environment.” *Journal of ACM*, pp. 604-632, New York, 1999.

[13] Lawrie, D. and Croft W. B. 2001. “Finding Topic Words for Hierarchical Summarization.” *In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, pages 349-357, 2001.

[14] Lent, B., Agrawal R., and Srikant R. 1997. “Discovering Trends in Text Databases.” *In Proceedings of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining (KDD'97)*, Newport Beach, California, August 1997.

[15] Leouski, A. V. and Croft W. B. 1996. *An Evaluation of Techniques for Clustering Search Results.* Technical Report IR-76, Department of Computer Science, University of Massachusetts, Amherst, 1996.

[16] Leuski, A. and Allan J. 2000. “Improving Interactive Retrieval by Combining Ranked List and Clustering.” *Proceedings of RIAO*, College de France, pp. 665-681, 2000.

[17] Liu, B., Chin C. W., and Ng, H. T. 2003. “Mining Topic-Specific Concepts and Definitions on the Web.” *In Proceedings of the Twelfth International World Wide Web Conference (WWW'03)*, Budapest, Hungary, 2003.

[18] Miller, D. H., Leek, T. and Schwartz, R. 1999. “A Hidden Markov Model Information Retrieval System.” *In SIGIR’99*, pp. 214-221.

[19] MSN search engine, 2010. http://search.msn.com/.

[20] Ponte, J. M. and W. B. Croft, 1998. “A Language Modeling Approach to Information Retrieval.” *In Proc. 21st Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR’98)*, pages 275-281, 1998.

[21] Ravichandran, Deepak and Eduard Hovy 2002. "Learning Surface Text Patterns for a Question Answering System", *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, ACL-2002, Philadelphia, July 2002, pp. 41-47.

[22] Singhal, A. 2001. “Modern Information Retrieval: a Brief Overview.” *IEEE Data Engineering Bulletin, Special Issue on Text and Databases*, 24(4), Dec. 2001, pp. 35-43.

[23] Soderland, Stephen. 1997. “Learning to Extract Text-based Information from the World Wide Web.” *In procee Knowledge Discovery and Data Mining (KDD-97)*, 1997, pp. 251-254.

[24] Song, F. and Croft, B., 1999. “A General Language Model for Information Retrieval.” *In: CIKM’99*, pp. 316–321.

[25] Srikant, R. and R. Agrawal, “Mining Generalized Association Rules,” *Proc. of the 21st VLDB Conference*, Zurich, Switzerland, pp.407-419, 1995.

[26] Srikanth, M. and Srikanth, R., 2002. “Biterm Language Models for Document Retrieval.” *In: SIGIR 2002*, pp. 425-426.

[27] Vivisimo clustering engine, 2004. http://vivisimo.com/.

[28] Voorhees, E. 2001. “Overview of the Question Answering Track.” Proceedings of the TREC-10 Conference. NIST, Gaithersburg, MD, 157–165.

[29] Wang, Ting-Ya and Sun Wu. 2003. “*Q & A in web search engine*.” M.S thesis. Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan, ROC., July 2003.

[30] Wu, S., Gais Home Page, http://gais.cs.ccu.edu.tw/

[31] Yahoo search engine, 2010. http://www.yahoo.com/.

[32] Zamir, O. and O. Etzioni. 1998. “Web Document Clustering: A Feasibility Demonstration,” *Proceedings of the 19th International ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'98)*, 46-54, 1998.

[33] Zamir, O. and O. Etzioni. 1999. “A Dynamic Clustering Interface to Web Search Results.” *In Proceedings of the Eighth International World Wide Web Conference (WWW8)*, Toronto, Canada, May 1999, pp. 1361 - 1374.

[34] Zeng, Hua-Jun, Qi-Cai He, Zheng Chen, Wei-Ying Ma, Jinwen Ma, 2004. “Learning to cluster web search results,” *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, July 25-29, 2004, Sheffield, United Kingdom.