Mining the Web for Answers to Natural Language Questions

Abstract

The web is now becoming one of the largest information and knowledge repositories. Many large scale search engines (Google, Fast, Northern Light, etc.) have emerged to help users find information. In this paper, we study how we can effectively use these existing search engines to mine the Web and discover the "correct" answers to factual natural language questions.

We propose a probabilistic algorithm called $QASM$ (Question Answering using Statistical Models) that learns the best query paraphrase of a natural language question. We validate our approach for both local and web search engines using questions from the TREC evaluation. We also show how this algorithm can be combined with another algorithm (AnSel) to produce precise answers to natural language questions.

1 Introduction

The web is now becoming one of the largest information and knowledge repositories. Many large scale search engines (Google, Fast, Northern Light, etc.) have emerged to help users find information. An analysis of the Excite corpus [7] of 2,477,283 actual user queries shows that around 8.4% of the queries are in the form of natural language questions. The breakdown of these questions is as follows: 43.9% can be counted as factual questions (e.g., "What is the country code for Belgium") while the rest are either procedural ("How do I ...") or "other" (e.g., syntactically incorrect). A significant portion of the 91.6% that are not in the form of natural language questions were still generated by the user with a question in mind.

Traditional information retrieval systems (including modern Web-based search engines as mentioned above) operate as follows: a user types in a query and the IR system returns a set of documents ordered by their expected relevance to the user query and, by extension, to the user's information need. This framework suffers from two problems: first, users are expected to follow a specific engine-dependent syntax to formulate their information need in the form of a query and second, only a small portion of each document may be relevant to the user query. Moreover, a study of search engine capabilities to return relevant documents [21] when the query is in the form of a natural language question

shows that search engines provide some limited form of processing of the question, namely removing stop words such as "who" and "where". Unfortunately, in factual question answering, such words can indicate the type of the answer sought and therefore simple removal may lead to lower accuracy of the search results.

We address these two problems in the context of factual, natural language question answering. In our scenario, when a user types in factual natural language questions such as "When did the Neanderthal man live?" or "Which Frenchman declined the Nobel Prize for Literature for ideological reasons?", he/she will expect to get back the precise answer to these questions rather than a set of documents that simply contain the same keywords as the questions. We make two assumptions here: one, users should not have to learn idiosyncratic search engine query syntax to ask a factual question; two, a document returned by a search engine may contain only a small portion that is relevant to the user question. It is therefore important to allow users to type questions as they see fit and only get the answer to their questions rather than the full document that contains it.

We introduce a probabilistic algorithm for domain-independent natural language factual question answering, $QASM$, which converts a natural language question into a search engine specific query. In our approach, we view question answering and the related problem of natural language document retrieval as instances of the noisy channel problem [3]. We assume that there exists a single best query Q that achieves high precision and recall given a particular information need, a particular search engine, and a particular document collection. The query Q is then transformed into a grammatical natural language question N through a noisy channel by the new proposed $QASM$ algorithm. Our goal is, given N, to recover the original query Q from the space U_Q of all possible queries that can be generated from N using a limited sequence of linguistically-justified transformation operators.

$QASM$ is based on expectation maximization (EM) and learns which paraphrase $\hat{Q}$ from U_Q achieves the highest score on a standardized benchmark. That paraphrase is then assumed to be the closest approximation to the original query Q. The algorithm makes use of a moderate number of labeled question-answer pairs for bootstrapping. Its generalization ability is based on the use of several classes of linguistic (lexico-semantic and collocational) knowledge.

We have incorporated $QASM$ in a system which is used to answer domain-independent natural language questions using both a local corpus and the Web as knowledge sources.

1.1 Question answering

The problem of factual question answering in the context of the TREC evaluation is best described in [24]. The goal is to return the most likely answers to a given question that can be located in a predefined, locally accessible corpus of news documents. To that end, most systems need to perform

the following steps: query analysis (to determine, for example, that a "who" question is looking for a person), document retrieval (to establish which documents are likely to contain the answer to the question), document analysis (to determine which portions of the document are relevant), and answer selection (return a short string containing the most likely correct answer from all retrieved documents). Note also that the questions used in the TREC evaluation are *guaranteed* to have at least one answer in the corpus, while in general no question is guaranteed to an answer on the Web.

The SMU systems, Lasso [17] and Falcon [10], make significant use of knowledge representation techniques such as semantic unification and abduction to retrieve relevant answers. For example, they are able to answer the question "Who was the first Russian astronaut to walk in space" by combining knowledge from the fact that the first *person* to walk in space was Leonov and from the fact that Leonov is Russian.

The IBM project [20, 22] requires that the corpus be tagged with semantic tokens called *QA-tokens*. For example, "In the Rocky Mountains" is tagged as PLACE$, while "The US Post Office" is tagged as ORG$. When a question is processed by the search engine, the QA tokens are passed along with the words in the query in order to enhance retrieval performance. This process requires that the entire corpus be pre-processed with QA-token information.

We claim that it is unrealistic to rely on semantic tagging or deep semantic understanding of both question and source documents when the QA problem is moved from a pre-defined, locally accessible corpus to the Web. It can be argued that a search engine may indeed annotate semantically all pages in its index, however that process is quite expensive in terms of processing time and storage requirements and it is clearly preferable to not have to rely on it. In other words, we are decoupling the problem of natural language document retrieval from the problem of answer selection once the documents have been found. In that case, only the top-ranked retrieved documents need to be annotated.

We need to note here that the service AskJeeves (www.ask.com) is not a QA system. It does process natural language queries but it returns sites that may be relevant to them without trying to identify the precise answers.

In this paper we will describe an algorithm for domain-independent factual question answering that does not rely on a locally-accessible corpus or to large-scale annotation. A system based on this algorithm thus provides the documents that contain the answers and needs to be connected to a component like AnSel to actually extract these answers. Note that AnSel only needs to annotate a small number of documents retrieved by $QASM$ and thus the problem of annotating the whole Web disappears.

Figure 1 shows some sample questions that our system can handle.

Note that we make several assumptions about the questions and their answers. First, we assume that each question may contain several answers (this assumption also appears in the TREC setup), all of

Question/Answer
Q: When did the Neanderthal man live?
A: about 125,000 to 30,000 years ago
Q: Which Frenchman declined the Nobel Prize for Literature for ideological reasons?
A: Jean-Paul Sartre
Q: What is the second largest planet in the Solar system
A: Saturn

Figure 1: Sample questions.

which are equally acceptable. For example, both "125,000 to 30,000 years ago" and "the Paleolithic age" can be considered to be correct answers to the question "When did the Neanderthal man live?". Second, for purposes of automatic evaluation, we assume that a document does not need to justify the answer it contains (this is a departure from TREC where all submissions are judged by hand as to whether they contain an instance of the correct answer that is justifiably the answer to the given question). As an example, if a document contains the string "the Paleolithic age" we will consider that it contains the answer to the question even if the document's topic is not about the Neanderthal man.

Let's now consider two alternative paraphrases of the same question `''When did the Neanderthal man live?''`. When we sent that question as it is to Excite, the first four hits didn't contain the answer. One of them was the home page of the Neanderthal museum in Germany while the second was about the Neanderthal flute. The third hit was about the discovery of the Neanderthal man but has no mention of the time he lived. The fourth one is an advertisement for a book on the Neanderthal man. Only the fifth match did contain a correct answer. On the other hand, when we used `live OR lived AND ''Neanderthal man''` as the query, the first two hits did contain accurate answers.

1.2 Statistical Translation Models

We will now describe a technique used in a large variety of text mining tasks - the statistical translation model. We make extensive use of this paradigm in our work.

A translation model is a probabilistic channel model. Given an observed string t in a *target* language T and a *source* language S, a probability $P(t|s)$ is assigned to all transformations from a string $s \in S$ to t. In statistical machine translation (e.g., French to English), for example, the probability $P(t|s)$ should be high for reasonable translations between the two languages, e.g., $P($`the small boy` $|$ `le petit garçon`$)$ should be high while $P($`the dog ate my homework` $|$ `le petit garçon`$)$ should be low.

The goal of a translation model is to find the t that maximizes the probability $P(s|t)$. In practice, the Bayesian rule is applied:

$$\underset{s}{argmax} P(s|t) = \underset{s}{argmax} P(s) * P(t|s)$$

and the goal becomes to find the value of s that maximizes $P(t|s)$.

The string s can be also thought as the *original* message that got somehow scrambled in the noisy channel and converted into the *target* string t. Then the goal of translation modeling becomes that of recovering s. Since the space of all possible strings in S is infinite, different heuristic techniques are normally used. We will discuss some of these techniques below.

Statistical translation models originate in speech processing (see [11] for an overview), where they are used to estimate the probability of an utterance given its phonetic representation. They have been also successfully used in part of speech tagging [5], machine translation [2, 4], information retrieval [3, 19], transliteration [12] and text summarization [13]. The reader can refer to [14] for a detailed description of statistical translation models in various applications.

In question answering, the source string is the query Q that produces the best results while the target string is the natural language question N. Obviously, there is an infinite number of queries that can be generated from a natural language question. These paraphrases are typically produced through a series of transformations. For example, the natural language question "Who wrote King Lear" can be converted into the query `(wrote | author) ''king lear''` by applying the following operators: (1) bracket "King" and "Lear" together to form an immutable phrase, (2) insert the word "author" as an alternative to "wrote", and (3) remove the question word "who" (note that before removing that word, the information that it conveys, namely the fact that the expected answer to the question is a person, is preserved through the use of the "insert" operator which adds a person word "author" to the query. The problem of question answering using a noisy channel model is reduced essentially to the problem of finding the best query $\hat{Q}$ which may have been generated from N. We will discuss in the next section what "best" means.

We have identified the following differences between statistical machine translation and question answering (QA).

1. In QA, the swap operator is not particularly needed as typical search engines give the same hits regardless of the order of the query terms.

2. Since swaps are not allowed, alignments are simpler in QA and the process of parameter estimation is simpler.

3. In QA, the source and target language are essentially the same language. The only difference is the addition of logical operators, parentheses, and double quotes in the queries.

4. The generation of queries in QA is much more robust than translation in the sense that the performance of a query typically degrades gracefully when an operator is applied to it while a correct translation can immediately become a horrible one when two words are swapped.

5. Since queries don't need to be grammatical English sentences, there is no need to use a language model (e.g., a bigram model) to determine the correctness of a query. On the other hand, this places an additional burden on the translation model, given that in SMT, the language model prevents a high-probability but ungrammatical translation from being produced.

6. SMT is trained on a parallel corpus with aligned sentences with hidden alignments. QA needs a parallel corpus of questions and answers while the actual queries that can produce documents containing the answers are hidden.

2 The $QASM$ algorithm

The $QASM$ algorithm (Question Answering using Language Modeling) is based on the premise that it is possible to select the best operator to apply on a particular natural language question (or query). That operator will produce a new query which is "better" than the one from which it was generated using some objective function such as precision or recall. We will later define some empirically justified objective functions to compare query paraphrases. In a sense the problem of finding the best operator given a query paraphrase is a classification problem. The class of a paraphrase is the same as the class of all equivalent paraphrases and is the operator that would most improve the value of the objective function when applied to the paraphrase. For example, the class of the paraphrase "who wrote King Lear" may be "delete-wh-word" if the resulting query "wrote King Lear" is the best of all possible queries generated by applying a single operator to "who wrote King Lear". Note that we make a distinction between the *composite* operator that converts a question to a query through a *series* of steps and the *atomic* operators that compose each step. For practical reasons, it is more feasible to deal with atomic operators when training the system.

In other words, we need to build a classifier which decides what operator is best applied on a given question N. In practice, we need to decompose the problem into a series of smaller problems and to produce a sequence of paraphrases Q_i with the following properties:

1. The first paraphrase Q_0 is the same as the natural language question N.

2. Each subsequent paraphrase Q_i is generated from Q_{i-1} using a single atomic operator. Note that operators have to be unambiguous given an arbitrary query. For example, "bracket" is ambigous,

because in a general query, many different subsequences can be bracketed together. On the other hand, "bracket the leftmost noun phrase in the query" is unambiguous.

3. If $F(Q_i)$ is the objective function that determines how good a paraphrase is (we will call this function the fitness function, borrowing a term from the evolutionary computation literature), then we want Q_i to be chosen from among all possible paraphrases $Q_{i,j}$ of Q_{i-1} using a single operator O_j, or that $Q_i \equiv Q_{i,k}$ for $k = \underset{j}{argmax} F(Q_{i,j})$.

4. The sequence of operators is interrupted when $k = \underset{j}{argmax} F(Q_{i,j})$ is the index of the identity operator I. The identity operator has the following property: $I(Q) = Q$. In other words, when no atomic operator can improve F, the process stops. Note that other stopping conditions (e.g., stability of the probability matrix) are also possible.

Two problems need to be resolved at this stage. First, it is obvious that the sequence of operators depends on the initial query $Q_0 \equiv N$. Since there is an infinite number of natural language questions N, it is necessary to perform some sort of smoothing. In other words, each question (and by extension, each query) has to be converted to a representation that preserves some subset of the properties of the question. The probability of applying a given operator on a question will depend on its representation, not on the question itself. This way, we avoid maintaining an operator probability distribution for each natural language question. In the following section, we will discuss a particular solution to the representation problem.

Second, we need to learn from a set of examples the optimal operator to apply given a particular paraphrase. The decomposition of transformation operators into atomic operators such as "insert" or "bracket" significantly reduces the complexity of finding the right operator to apply on a given question.

Since it is very expensive to produce a large training corpus of pairs of questions and their best paraphrases, we have to recur to an algorithm that is stable with regard to missing data. Such an algorithm is the expectation maximization (EM) algorithm.

2.1 The EM algorithm

The EM algorithm [6] is an iterative algorithm for maximum likelihood estimation. It is used when certain values of the training data are missing. In our case, the missing values are the paraphrases that produce the best answers for given natural language questions. We only have question-answer pairs but no paraphrases. In other words, the known variables are the scores for each operator; the hidden variables are the probabilities for picking each operator.

The EM algorithm uses all the data available to estimate the values of the missing parameters. Then it uses the estimated values to improve its current model. In other words, the EM algorithm works as

follows: first, it seeds the parameters of the model with some reasonable values (e.g., according to the uniform distribution). Then it performs the following two steps repeatedly until a local maximum has been reached.

- E-step: use the best available current classifier to classify some datapoints
- M-step: modify the classifier based on the classes produced by the E-step.

The theory behind EM [6] shows that such an algorithm is guaranteed to produce increasingly better models and eventually reach a local maximum.

2.2 Generic operators

We now need operators that satisfy the following criteria: they must be easy to implement, they must be unambiguous, they must be empirically justified, and they must be implemented by a large number of search engines. A list of such generic operators follows (see also Figure 2). We call them generic because they are not written with any particular search engine in mind. In the following section, we will discuss how some of these operators can be operationalized in a real system.

1. INSERT, add a word or phrase to the query (similar to the fertility operator in SMT),
2. DELETE, remove a word or phrase ("infertility" operator),
3. DISJUNCT, add a set of words or phrases in the form of a disjunction,
4. REPLACE, replace a word or phrase with another,
5. BRACKET, add double quotes around a phrase,
6. REQUIRE, insist that a word or a phrase should appear (most search engines require such operator to explicitly include a word in the query),
7. IGNORE, for example the query `cleveland -ohio` will return documents about President Cleveland and not about the city of Cleveland, Ohio.
8. SWAP, change the order of words or phrases,
9. STOP, this is the identity operator,
10. REPEAT, add another copy of the same word or phrase

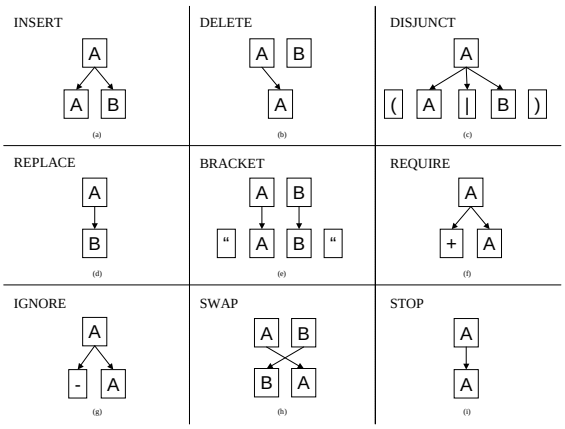

Figure 2: Sample operators.

2.3 The probabilistic generation model

We will now turn to the need for question and query representations. The space of questions is infinite and therefore any classification algorithm must use a compact representation of the questions. An empirical analysis [21, 23] shows that certain features of questions interact with the scores obtained from the search engines and that questions with the same features tend to be treated similarly. In the next section we will discuss the particular features that we have implemented. At this moment, we only want to specify that all questions with the same feature representation will be treated in the same way by our algorithm. We will call the state of all questions with the same values for all features the *context* C of the query.

The model Θ contains the probabilities $p(O_i|C_J)$ of applying operator O_i given context C_j are represented in a two-dimensional probability matrix. Given a state (context) we can determine the most likely operator. Later in this section we will discuss our algorithm for learning the values of the Θ matrix. Note that Θ has N_s states and N_o operators and the sum of operator probabilities for each row (state) is: $\sum_i^{N_o} p(O_i|C_j) = 1$.

2.4 Generating query paraphrases

In order to learn the Θ matrix, we need to be able to produce different paraphrases of a given natural language question N. We will N as the initial seed query Q_0: $Q_0 \equiv N_i$ and then apply $QASM$ to produce subsequent paraphrases Q_i from Q_{i-1}.

2.5 Evaluating paraphrase strength

At each iteration, we must determine which operator is the best. We need to define a *fitness* function. In general, such fitness function can be one of a large number of information theoretic measures such

as entropy or perplexity. It can also be a metric from information retrieval such as precision (accuracy) or recall (coverage). In the next section, we will introduce $TRDR$, a metric that we found particularly useful.

2.6 The $QASM$ algorithm

We will now introduce the $QASM$ algorithm. It is shown in Algorithm 1. Some notational clarifications: C_j represents the context (or state) of a question. Θ is the probabilistic model that determines the probability of applying a given operator on question that is represented in a particular state. The initialization step shown in the Figure can be replaced with one that uses an appropriate prior distribution other than the uniform distribution. After each iteration of the EM algorithm, the query is modified using an operator generated probabilistically from the current distribution for the state where the current paraphrase belongs. The global stop criterion is a function of the change in the probability matrix Θ.

Algorithm 1 $QASM$: Learning algorithm

initialize Θ with $\forall i, p(O_i|C_j) = 1/N_o$
set $Q_0 \equiv N$; set $j = 0$
repeat
  extract documents that match Q_j
  compute paraphrase fitness $F(O_i|C_j)$ for all i
  let $l = \underset{k}{argmax} F(O_k|C_j)$
  if $F(STOP|C_j) \geq F(O_k|C_j)$ **then**
    next iteration
  end if
  pick an operator O_m according to Θ.
  recompute context C_{j+1}
  rerank operators based on fitness
  readjust/normalize Θ based on the reranking
  apply O_m on Q_j to produce Q_{j+1}
  increment j
until $\delta(\Theta) < \varepsilon$

2.7 Decoding algorithm

Once the probabilistic model P has been trained, it can be used to process unseen questions. The decoding algorithm (to borrow another term from SMT) is shown in Algorithm 2

Algorithm 2 Decoding algorithm

Θ is given from the training stage
set $Q_0 \equiv N$; set $j = 0$
repeat
  let $l = \underset{k}{argmax} F(O_k|C_j)$
  apply O_l on Q_j to produce Q_{j+1}
  increment j
until $F(STOP|C_{j-1}) \geq F(O_k|C_{j-1})$

3 Implementation and Experiments

We now describe the operationalization of the $QASM$ algorithm in our system. We will address the following issues: choice of operators, training and test corpora of questions and answers, choice of document corpora and search engine, question preprocessing, etc.

3.1 Specific operators

We have chosen a "basis" composed of 15 operators, grouped into four categories: DELETE, REPLACE, DISJUNCT, and OTHER. We use five DELETE operators (delete all prepositions, delete all wh-words, delete all articles, delete all auxiliaries, and delete all *other* stop words based on a list of 163 stop words). We also use four REPLACE operators (replace the first noun phrase with another, replace the second noun phrase, replace the third noun phrase, and replace the first verb phrase in the question). Note that by specifying which exact terms we will replace, we are addressing the "where" question. The four DISJUNCT operators are similar to the REPLACE operators, except that the new words and phrases are disjoined using OR statements. Finally, the IDENTITY operator doesn't modify the query.

When deciding *what* word to insert with the REPLACE and DISJUNCT operators, we use two alternative approaches. In the first case, we look at the first three synonyms and the closest hypernym of the first sense of the word (in the correct part of speech: verb or noun) based on the WordNet database [16]. The second case uses distributional clusters of fixed size of the words [18]. We don't have room here to describe the implementation of the distributional algorithm but here is the basic idea: if two words appear in the same context (looking at words in a window) in a large corpus, they are considered to be distributionally similar. As an example, "professor" and "student" are not distributionally similar because they tend to appear together, while "book" and "essay" are since they are in complementary distribution in text. Figures 3 and 4 show up to five words or phrases similar to the word in the left column.

Note that we did at this point, our implementation doesn't include the following operators: IG-

word	related words
book	publication, product, fact, dramatic composition, record
computer	machine, expert, calculator, reckoner, figurer
fruit	reproductive structure, consequence, product, bear
politician	leader, schemer
newspaper	press, publisher, product, paper, newsprint

Figure 3: Related words produced by WordNet.

word	related words
book	autobiography, essay, biography, memoirs, novels
computer	adobe, computing, computers, developed, hardware
fruit	leafy, canned, fruits, flowers, grapes
politician	activist, campaigner, politicians, intellectuals, journalist
newspaper	daily, globe, newspapers, newsday, paper

Figure 4: Related words produced by distributional similarity.

NORE, REPEAT, and SWAP. We implemented BRACKET in the preprocessing stage.

3.2 Fitness functions

Instead of precision and recall, we use total reciprocal document rank ($TRDR$). For each paraphrase, the value of $TRDR$ is the sum of the reciprocal values of the rank of all correct documents among the top 40 extracted by the system: $TRDR = \frac{1}{n}(\sum_i^n \frac{1}{rank_i})$. For example, if the system has retrieved 10 documents, of which three: the second, eighth, and tenth, contain the correct answer, $TRDR$ for that given paraphrase is $\frac{1}{2} + \frac{1}{8} + \frac{1}{10} = .725$.

The metric used in TREC is slightly different from ours. In TREC's case, only the rank of the best answer counts so if the correct answer is retrieved in first place only, the score will be higher than when the correct answer appears in all places from second to fortieth. Several observations motivated this switch in formulas: (1) Since we are looking at finer distinctions between related paraphrases of the same question, we need to pay more attention to subtle differences in performance, (2) In a Web environment, some URLs on the hit list may no longer be around when the system tries to retrieve them, so we want to reward systems, which other things being equal, return more documents with answers, and (3) the need to accept different variants of the same answer, as illustrated in the following example.

Figure 5 gives a feeling for the diversity of the wordings of answers to the same question on the Web and in some way provide an additional justification for our choice to include all correct hits in TRDR. In this example, three competing documents specify the place where Dylan Thomas died.

We should note that the highest score that a given paraphrase can obtain on a given question is 4.279 (the sum of $1/n$ for n from 1 to 40). Qualitatively, a paraphrase is acceptable if it obtains a score of at

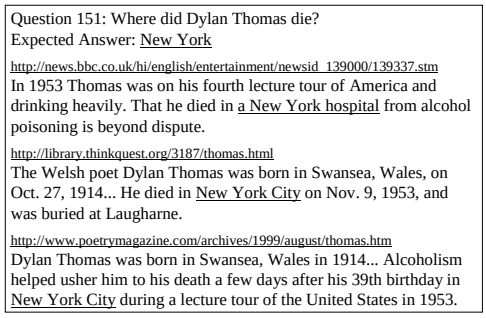

Figure 5: Sample texts answering a question about Dylan Thomas.

least .300 which corresponds roughly to one hit among the top five or two hits in the next five.

3.3 Training data and choice of corpora and search engines

To train our system, we used pairs $\{N, A\}$ collected from a variety of sources: the TREC8 and TREC9 collections (893 datapoints) plus our own collection of 2,224 datapoints.

We used zzsearch (a search engine developed by our group) on the local corpus of 240,000 AP news articles from the TREC collection. On the Web, we intended to use Google as the primary search engine, however at some point it stopped processing queries longer than 10 words so we had to switch to the Fast search engine [1]. We built a module that converts the same underlying query to the syntax of these two search engines.

3.4 Preprocessing

Before we run any queries on a search engine, we preprocess them in order to identify immutable phrases that would be preserved in the paraphrasing process. We use the ltchunk utility [15] to determine noun phrases and we filter out these that contain an initial determiner followed by a single word such as “the man”.

3.5 Feature representations of queries

We decided to adopt the particular representation scheme proposed by Radev et al. [21, 23] using three features: semantic type of question based on *wh*-word (e.g., PERSON, LOCATION, etc.), number of words in query, number of proper nouns in query.

As an example, `who wrote King Lear` can be represented as (“PERSON”,4,1). The generalizations based on the representation can capture such patterns as “the probability of applying a DELETE operator on a shorter query is smaller than on a longer query”.

3.6 Sample run

We will now run through an example of the system in action. We will show how the $QASM$ algorithm readjusts the values of the Θ model using a particular question from the TREC 8 corpus. The fitness scores are computed using the Fast search engine.

In this example, the question N is `Which country is the biggest producer of tungsten` (that is question 14 from TREC8). The expected answer is "China".

The initial paraphrase $Q_0 \equiv N$ is relegated to state ("LOCATION",8,0) since it is a "where" question, it is 8 words in length, and contains no proper nouns.

The probability values for the row ("LOCATION",8,0) are all set to be 1/15 = .667 under the default assumption that all 15 operators are equally likely.

Figure 6 shows the $TRDR$ values for the first 8 paraphrases of N. The 8 operators shown are the IDENTITY operator, all five DELETE operators, and the first two REPLACE operators (see Subsection 3.1). In this run, we use the variants of the operators based on WordNet and not on distributional clustering.

No.	**Operator**	**Paraphrase**	$TRDR$
0	IDENTITY	What country is the " biggest producer " of tungsten	0.837
1	DEL_WH	country is the " biggest producer " of tungsten	1.755
2	DEL_AUX	What country the " biggest producer " of tungsten	1.322
3	DEL_ART	What country is " biggest producer " of tungsten	1.436
4	DEL_PREP	What country is the " biggest producer " tungsten	1.436
5	DEL_STOP	What country is the " biggest producer " of tungsten	0.837
6	REPL_1N	What (" administrative district " OR " political unit " OR people OR region OR " geographical area " OR " rural area ") is the " biggest producer " of tungsten	1.181
7	REPL_2N	What country is the "biggest producer " of (" metallic element " OR wolfram OR w OR " atomic number 74 ")	1.419

Figure 6: Example of the operation of $QASM$.

Once all fitness scores are computed, the probabilities are readjusted proportionally to the scores. As a result, the probability for operators 1 and 3 increase at the expense of operators 0, 5, and 6. The resulting probabilities for the first 8 operators are as follows (only the first eight are shown for lack of space): 0.0536, 0.1123, 0.0846, 0.0919, 0.0919, 0.0536, 0.0756, and 0.0908.

The next paraphrase is generated according to the new probability distribution. Obviously, at this stage, all operators still have a chance of being selected as no fitness score was zero. In this example, operator 0 (IDENTITY) was selected.

The new state is the same as the previous one: ("LOCATION",8,0). The probability distribution has now changed. Since the values of $TRDR$ in the second iteration will be the same as in the first,

the probabilities will be readjusted once more in the same proportion as after the first iteration. After the second EM iteration, the probabilities for state ("LOCATION",8,0) are as follows: 0.0374, 0.1643, 0.0932, 0.1100, 0.1100, 0.0374, 0.0745, and 0.1074. After five iterations in the same state they become 0.0085, 0.3421, 0.0830, 0.1255, 0.1255, 0.0085, 0.0473, and 0.1184. After 23 iterations in the same state, the probability of applying operator 2 is 0.9673 while all 14 other probabilities tend to 0.

If we allow QASM to pick each subsequent operator according to Θ we observe that the sequence that achieves the highest score for the state ("LOCATION",8,0) is to apply operator 2 followed by operator 4. Note that the state of the paraphrase changes after operator 2 is applied as one word has been removed.

After all probabilities are learned from the training corpus of 2,224 datapoints, $QASM$ can proceed to unseen questions. For lack of space, we omit a complete illustration of the decoding process. We will just take a look at all questions that fall in the same class as the one in the example. There are 18 such questions in our test corpus and for 14 of them (77.8%), as predicted, the sequence of operators 2 and 4 is the one that achieves the highest score. In two other cases, this sequence is second best and in the last two cases it is still within 20% of the performance of the best sequence. For the 18 questions the performance over the baseline (sending the natural language question directly to the search engine) goes from 1.31 to 1.86 (an increase of 42.0%).

4 Related Work

There has been a lot of effort in applying the notion of language modeling and its variations to other problems. For example, Ponte and Croft [19] adopt a language modeling approach to information retrieval. They argue that much of the difficulty for IR lies in the lack of an adequate indexing model. Instead of making prior parametric assumptions about the similarity of documents, they propose a non-parametric approach to retrieval based probabilistic language modeling. Empirically, their approach significantly outperforms traditional tf*idf weighting on two different collections and query sets.

Berger and Lafferty [3] suggest a similar probabilistic approach to information retrieval based on the ideas and methods of statistical machine translation. The central ingredient in their approach is a noisy-channel model of how a user might "translate" a given document into a query. To assess the relevance of a document to a user's query, they estimate the probability the query would have been generated as a translation of the document, and factor in the user's general preferences in the form of a prior distribution over documents. They propose a simple, well-motivated model of the document-to-query translation process, and describe the EM algorithm for learning the parameters of this model in an unsupervised manner from a collection of documents.

Brown et al. [4] lay out the mathematical foundation of statistical machine translation, while Berger et al. [2] presents an overview of Candide, a system that uses probabilistic methods to automatically translate French text into English.

Glover et al. [8, 9] address the issue of query modification when searching for specific types of Web pages such as personal pages, conference calls for papers, and product announcements. They employ support vector machines to learn engine-specific words and phrases that can be added to a query to locate particular types of pages. Some of the words are quite unobvious, for example adding "w" as a word to a query improves the performance of their system, Inquirus2, when it is set to search for personal pages.

Dempster, Laird, and Rubin [6] are the first ones to formalize the EM algorithm as an estimation technique for problems with incomplete data.

5 Conclusion

In this paper we described an algorithm for learning a sequence of transformations that need to be applied on a natural language question so that it can be run on a search engine and retrieve documents containing the answer to the question. Our algorithm, $QASM$, is based on the EM estimation algorithm and employs linguistically justified transformational operators. $QASM$ has been implemented in a natural language question answering system with two different backends - one based on a local corpus and another - using an external search engine which retrieves answers from the World-Wide Web.

Our two main contributions are (a) the use of the Web as a knowledge source for domain-independent question answering and (b) the $QASM$ algorithm which produces the paraphrase of a natural language question that is most likely to produce a list of hits containing the answer(s) to the question.

5.1 Future work

We plan to link $QASM$ with the AnSel system to complete the pipeline question - documents - set of potential answers. We also intend to study user-specific question answering. For example, the answer to the question "where is the Dynasty restaurant" may be different based on the user's location, experience, and preferences. We plan also to investigate the problem of cross-lingual question answering where the question is in one language and the answer - in another. Finally, we are pursuing a separate study that will indicate whether the choice of search engine can be parameterized in the probabilistic model. If we are successful in this endeavor, we will be able to build a meta-search engine which will both choose the best paraphrase of a question and the search engine where to send it.

References

[1] The Fast search engine. http://www.alltheweb.com, 2001.

[2] A. Berger, P. Brown, S. Pietra, V. Pietra, J. Lafferty, H. Printz, and L. Ures. The candide system for machine translation. In *Proceedings of the ARPA Conference on Human Language Technology, 1994.*, 1994.

[3] A. Berger and J. Lafferty. Information retrieval as statistical translation. In *Proceedings, 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, California, August 1999.

[4] P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[5] K. Church. A stochastic parts program and a noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, Texas, 1988.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society series B*, 39:1–38, 1977.

[7] The Excite query corpus. ftp://ftp.excite.com/pub/jack/Excite_Log_12201999.gz, 1999.

[8] E. Glover, G. Flake, S. Lawrence, W. Birmingham, and A. Kruger. Improving category specific web search by learning query modifications. In *Symposium on Applications and the Internet*, Jan 8-12 2001.

[9] E. J. Glover, S. Lawrence, M. D. Gordon, W. P. Birmingham, and C. L. Giles. Web search – your way. *Communications of the ACM (accepted)*, 2001.

[10] S. Harabagiu, D. Moldovan, M. Paşca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Gîrju, V. Rus, and P. Morărescu. The TREC-9 question answering track evaluation. In *Text Retrieval Conference TREC-9*, Gaithersburg, MD, 2001.

[11] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, Massachusetts, 1997.

[12] K. Knight and J. Graehl. Machine transliteration. *Computational Linguistics*, 24(4), 1998.

[13] K. Knight and D. Marcu. Statistics-based summarization – step one: sentence compression. In *Proceedings, Seventeenth Annual Conference of the American Association for Artificial Intelligence*, Austin, Texas, August 2000.

[14] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[15] A. Mikheev. Tagging sentence boundaries. In *Proceedings, SIGIR 2000*, 2000.

[16] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography (special issue)*, 3(4):235–312, 1990.

[17] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus. The structure and performance of an open-domain question answering system. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hong Kong, October 2000.

[18] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *30th Annual Meeting of the ACL*, pages 183–190, 1993.

[19] J. Ponte and B. Croft. A language modeling approach to information retrieval. In *Proceedings, 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia, August 1998.

[20] J. Prager, E. Brown, A. Coden, and D. Radev. Question-answering by predictive annotation. In *Proceedings, 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, July 2000.

[21] D. R. Radev, K. Libner, and W. Fan. An empirical evaluation of the capability of state-of-the-art search engines to answer natural language questions. Submitted, 2001.

[22] D. R. Radev, J. Prager, and V. Samn. Ranking potential answers to natural language questions. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, WA, May 2000.

[23] D. R. Radev, H. Qi, and W. Fan. Query modulation in web-based question answering. Submitted, 2001.

[24] E. Voorhees and D. Tice. The TREC-8 question answering track evaluation. In *Text Retrieval Conference TREC-8*, Gaithersburg, MD, 2000.