Question Answering with Discriminative Learning Algorithms

Junlan Feng

AT&T Labs – Research
180 Park Avenue, Florham Park, NJ, 07932 - USA
junlan@research.att.com

Abstract

In this paper, we describe a discriminative learning approach for question answering. Our training corpus consists of 2 million Frequently Asked Questions (FAQs) and their corresponding answers that we mined from the World Wide Web. This corpus is used to train the lexical and semantic association model between questions and answers. We evaluate our approach on two question answering tasks: 2003 Text Retrieval Conference Question Answering task, and finding answers to FAQs. In both cases, the proposed approach achieved significant improvements over the results for an information retrieval based question answering model.

Index Terms: question answering (QA)

1. Introduction

Question answering (QA) is an interactive human-machine process that aims to find a direct answer to a natural language question from a collection of documents. Unlike state-of-art spoken dialog systems, which are often configured by a hand-crafted dialog flow and designed for completing tens to hundreds of user requests, QA systems are controlled by information provided in unstructured documents and designed to answer natural language requests pertaining to the content of the given documents. The document collection could be the World Wide Web [1], an enterprise website [2], a set of prepackaged responses [5], or millions of newswire articles [3].

A typical diagram of modern QA systems [4] is illustrated in Figure 1. A QA system takes a question as input and returns answers from given documents. Question analysis consists of question preprocessing, question classification, and query expansion. Question preprocessing parses a natural language question with syntactic tags such as part-of-speech tags and named entity (NE) tags. Question classification determines the expected answer type of the question that allows subsequent processes to precisely locate and verify the answer. Query expansion formulates the keyword queries for being used to retrieve documents. Information retrieval (IR) retrieves documents relevant to the generated keyword queries. These retrieved documents are deemed likely to contain valid answers to the question. Answer extraction extracts the precise answer from the retrieved documents. The underlying techniques to support these steps are built upon language processing technologies, IR and statistics. A massive number of knowledge-intensive approaches have been developed for QA. For instance, the LCC system, PowerAnswer, transforms questions and possible answer text into a logic representation and then builds a formal proof for valid answers [12]. Data driven approaches are relatively less explored, yet recently have received much more attention. Li et al. [6] described a learning approach to classifying questions into two-layered fine-grained classes. A.Berger et al. [5] and R. Soricut et al. [6] used statistical translation models [7] to learn the lexical relationship between questions and answers.

In this paper, we describe a new learning approach for training the QA model from pre-answered questions. We built a training corpus consisting of 2 million Frequently Asked Questions (FAQs) and their corresponding answers. We collected these FAQ-Answer pairs by mining the World Wide Web. We evaluated our approach using two types of QA data, namely, 2003 Text Retrieval Conference (TREC) QA test data [3], and a collection of FAQ-Answer pairs.

The rest of this paper proceeds as follows. Section 2 discusses the main challenges for QA, reviews related work and describes our proposed approach. In section 3, we present the experimental results and analysis. Section 4 provides our summary and conclusion.

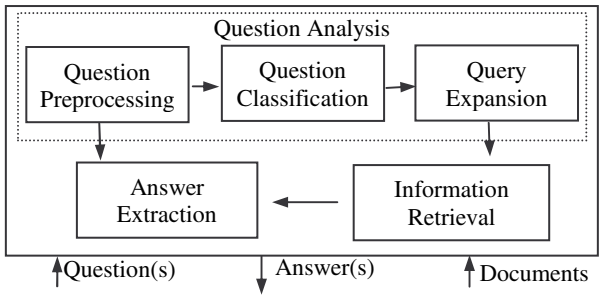

Figure 1: A typical diagram of modern QA systems

2. Learning Approaches to Question Answering

2.1 Motivation

As Figure 1 shows, most QA systems rely on IR to return top relevant documents, from which the precise answer is extracted. In this paper, we assume candidate responses can be prepackaged. For our training data, all the answers are originally individual units. Section 3 describes how we prepackage answers with the test data. Under this assumption, a natural simple solution for QA is to use the IR model, where the match score between a question and an answer is based on exact keyword match. The candidate response achieving the highest score is returned as the best answer. For IR, each keyword is typically weighted using the *tf.idf* mechanism [8]. Given a collection of prepackaged answers $A = \{a_1, a_2, ...a_N\}$, the *tf.idf* weights for each word w_j appearing in a_i can be represented as:

$$tf.idf(a_i, w_j) = tf_{ij} * idf_j = tf_{ij} \cdot \log_2 \frac{N}{df_j} \qquad (1)$$

where tf_{ij} is referred to as *term frequency* - the frequency of the word w_j in the answer a_i ; df_j is referred to as *document frequency* - the number of answers in which w_j occurs at least once; and *N* is the total number of answers in the collection. The QA match score based on *tf.idf* between an m-word question $q = \{v_1, v_2, ...v_m\}$ and an n-word answer $a = \{u_1, u_2, ...u_n\}$ is represented as:

$$score(q,a) = \sum_{w \in q,a} tf.idf(a,w) \cdot tf.idf(q,w) \cdot Z(q,a) \quad (2)$$

where $Z(q,a)$ is a normalization factor:

$$Z(q,a) = \frac{1}{\sqrt{\sum_{w \in q}(tf.idf(q,w))^2} \cdot \sqrt{\sum_{w \in a}(tf.idf(a,w))^2}} \quad (3)$$

This *tf.idf* based match model has been successfully used in the search engines. However, as the experiments will show in this paper, it performs poorly in QA systems. This poor performance results from the following QA challenges:

- *tf.idf* weights are solely determined by the answer collection. However, some words, in general, are more important for answer retrieval. For instance, the word *"china"* is more likely to be a QA common word than the word *"take"* independent of their occurrences in the specific answer set.
- There is a semantic gap between questions and answers. A question expresses an information need that the valid answer is expected to satisfy. For example, a *"when"* question often expects an answer containing a *TIME /DATE* named entity value. A *"why"* question expects a reason for the matters concerned. The IR model based on exact word match doesn't provide a solution to bridge this QA semantic chasm.
- Questions and answers are often phrased in different vocabularies and styles. We observed that 12% of the QA pairs in our training corpus don't have a single common keyword. This mismatch results in many cases that the correct answer is not retrieved or ranked as the first. Researchers have addressed this challenge with various methods such as query expansion [9] and statistical translation models [5][6]. Query expansion expands the keywords appearing in the question to a bigger set of words that are likely to appear in the answer. For instance, "reach" can be expanded to "fax" and "phone". The difficulty is that the expanded word such as "fax' may bring in noise and twist the user's information needs for some cases. [5] and [6] proposed more statistical algorithms including latent variable models and statistical translation models to bridge the lexical gap between questions and answers. From the experiments reported in [5], the highly parameterized techniques based on statistical translation models showed the best performance. The learned translation models were applied to re-rank tens of candidate answers/documents returned by IR. However, it's hard to scale up this algorithm to directly select answers from a large collection of documents.

In the following, we describe a discriminative learning approach to address these challenges. Our training objective is to optimize the importance weights of the query words, and to learn the lexical and semantic relationship between questions and answers.

2.2 Perceptron

We begin with IR as our baseline, where the match score between a question q and an answer a is calculated with the formula (2). The best answer for a given q_i is selected as:

$$a_i = \arg\max_{a_j \in A} score(q_i, a_j) \quad (4)$$

where $A = \{a_1, a_2, ...a_N\}$ is a collection prepackaged answers as defined in Section 2.1. We rewrite the formula (2) with a general linear representation:

$$score(q,a) = \phi(q,a) \cdot W \quad (5)$$

where ϕ is a function that maps a question q and an answer a to a k dimensional feature vector $\phi(q,a) \in \Re^k$; $W \in \Re^k$ is a k dimensional parameter vector. For the *tf.idf* based match score given by (2), k is the number of keywords recognized in the answer collection; the feature vector is constituted by the *tf.idf* weights:

$\phi(q,a) = (docf(w_1,q,a), docf(w_2,q,a), ..., docf(w_K,q,a))$,

where $docf$ is the normalized *tf.idf* weight defined as follows:

$$docf(w,q,a) = \begin{cases} tf.idf(a,w) \cdot tf.idf(q,w) \cdot Z(q,a), & w \in q,a \\ 0 & otherwise \end{cases} \quad (6)$$

W in this case is a k dimension vector with all components equal to one.

Given this linear representation, we propose to use the voted perceptron algorithm[10] to estimate the parameter vector W for any given feature function ϕ. The learning objective is to achieve higher answer accuracy, which is defined as:

$$accuracy = \frac{\text{number of times the correct answer is ranked first}}{\text{total number of questions in the test set}}$$

We now describe the perceptron training algorithm. We will elaborate on a series of methods for the feature function $\phi(q,a)$ in Section 2.3. The voted perceptron algorithm described in [10] has been applied to various natural language processing problems such as tagging or parsing. It has been shown on these tasks to be competitive compared to modern learning algorithms including conditional random fields and support vector machines [10]. We make an attempt in this paper to apply this method to QA. We give a variant of the voted perceptron training algorithm in Figure 2 for our QA problem. The algorithm takes *T* passes over the training examples. Parameters in *W* are initialized differently for different types of features, which we will describe in Section 2.3. For each question q_i in the training corpus, a temporal parameter vector W' is created as a copy of *W* and the best answer $\tilde{a}_i$, from the same website that the question appears on, is returned using the current *W*. If the returned best answer is not correct, the temporal parameters W' are updated in a simple additive fashion. Parameters *W* are updated using the average parameters at the end of each training pass. The parameter vector *W* is the output of the learning process after the *T'th* iteration. Note that in our training process, for each training question, we limit the answer search space to the website where the question originated. We set this limitation because in our training corpus we observed that for a number of cases, the semantically same question appears on multiple websites and has been answered differently. Consequently, it is difficult to automatically judge the correctness of a returned

answer if it is from a website different than the one the question originated. Within a given website, it is more reasonable to assume that the associated answer for a question is the unique correct answer.

Inputs: A training set of question-answer (QA) pairs $\{(q_i, a_i)\}$ for $i = 1...n.$, where each QA pair has an associated website *site*(*i*) that (q_i, a_i) appears on; A parameter *T* specifying the number of iterations over the training set; A feature function $\phi(q,a) \in \Re^k$ that maps a question-answer pair to a *k*-dimensional feature vector.

Initialization: Initialize the parameter vector *W*

Training:

For *t=1…T,*

Set a k-dimensional vector: $W^{sum} = 0$

For i=1…n

- Create a temporal parameter vector: $W' = W$
- Find the best answer using the current *W*: $\tilde{a}_i = \arg\max_{site(j)=site(i)} \phi(q_i, a_j) \cdot W$
- If $\tilde{a}_i \neq a_i$, then update the W': $W' = W' + \phi(q_i, a_i) - \phi(q_i, \tilde{a}_i)$
- Accumulate Parameters: $W^{sum} = W^{sum} + W'$

Average Parameters: $W = W^{sum} / n$

Output: *W*

Figure 2: A variant of the perceptron algorithm for QA

2.3 Features

This section discusses several increasingly sophisticated features that we developed towards bridging the lexical and semantic gap between question language and answer language.

Feature Set I: *Exact Word Match Features*. We begin with using normalized *tf.idf* weights *docf* (defined in (6)) as features. The QA match score (5) becomes:

$$score(q,a) = \phi(q,a) \cdot W = \sum_{w \in q,a} docf(w,q,a) \cdot qf(w) \quad (7)$$

$qf(w)$ is the parameter for characterizing the importance of the query keyword *w*. In our experiments, $qf(w)$ is initialized to 1 and is estimated using the perceptron algorithm.

Feature Set II: *Semantic Correlation Features*. Most questions start with a question phrase such as "how much does' and "what country". As Agichitain defined in [8], a question phrase, denoted as *qp*, is a stream of text at the beginning of the question. A question phrase is used to address the specific information need and is often not repeated in the corresponding answers. Modern QA systems often include a question classification component (as shown in Figure 1), that classifies a question into an NE type [11], for example, classifying a *"when*" question to the *TIME/DATE* category. An instance of this NE category is expected to be included in the answer. This would allow subsequent processes to precisely locate and verify the answer. Instead of training such a question classifier from human labeled data, we propose an approach to statistically associating a question phrase with multiple NE tags, which are tagged in the answers. We limit a question phrase to be a text string with no more than four words. In our experiments, we consider a question phrase only when its frequency in our training questions is greater than a preset threshold *30*. For each question phrase *qp*, we associate it with a NE *ne* by calculating mutual information:

$$I(qp,ne) = H(p(ne \in a)) - p(qp \in q)H(p(ne \in a \mid qp \in q)) - p(qp \notin q)H(p(ne \in a \mid qp \notin q))$$

where H(.) is the entropy function :

$$H(p) = -p\log_2(p) - (1-p)\log_2(1-p)$$

Based on the value of the mutual information *I(qp,ne)* calculated from the training data, we build a *ne* set for each *qp*, represented as *NE(qp)*. For instance, *"MONEY"* and *"NUMBER"* are the NEs selected for the *qp* "what *scholarship*". Each training question *q* is then reformulated using the following procedure:

(1) *Find the longest qp that q begins with*
(2) *Locate the nouns in qp*
(3) *Replace qp with its associated NE(qp) tags and the nouns appearing in qp*

With this reformulation, we extend the QA match score (7) to be the following:

$$score(q,a) = \phi(q,a) \cdot W = \sum_{w \in q,a} docf(w,q,a) \cdot qf(w) + \sum_{ne \in NE(qp), qp \in q} docf(ne,q,a) \cdot \lambda_{sem}(qp,ne) \quad (8)$$

Where $\lambda_{sem}(qp,ne)$ is the association parameter between a question phrase *qp* and an NE tag *ne*. $\lambda_{sem}(qp,ne)$ is initialized to 0 in our experiments and is estimated with the perceptron training algorithm.

Feature Set III: *Lexical Association Features.* We are inspired to learn the QA lexical relationship by the following observations. First, over 60% of the questions in our training data are open-ended (such as how, why, yes/no types of questions). The information these questions ask for is more complicated than simple NE values. As a result, the QA semantic gap can hardly be bridged through associating question phrases and NE tags for open-ended questions. Second, questions and answers are often phrased using different vocabularies and different styles. Third, answers for open-ended questions are usually much longer than the question. Hence, there is a higher chance that words in the answer are not included in the question.

Similar to the above method that we used to find the associated NE tags for question phrases, we find associated answer words *{v}* for each query word *w* by calculating mutual information:

$$I(w,v) = H(p(v \in a)) - p(w \in q)H(p(v \in a \mid w \in q)) - p(w \notin q)H(p(v \in a \mid w \notin q))$$

Based on the value of *I(w,v)*, we build an answer word expansion set for each *w*, represented by *Exp(w)*. For instance, for the word *"travel"* we achieved the expansion set: *Exp(travel)= {trip, airline, flight, ticket, traveler}.*

With these expansion sets, we extend the QA match score (8) to:

$$score(q,a) = \phi(q,a) \cdot W = \sum_{w \in q,a} docf(w,q,a) \cdot qf(w) + \sum_{ne \in NE(qp), qp \in q} docf(ne,q,a) \cdot \lambda_{sem}(qp,ne) + \sum_{v \in Exp(w), w \in q, v \in a} docf(v,q,a) \cdot \lambda_{lex}(w,v) \quad (9)$$

Where $\lambda_{lex}(w,v)$ is the lexical association parameter. We initialize $\lambda_{lex}(w,v)$ to 0 during training. Experimental results with the above three types of features will be reported in the

next section. The learned parameters including *qf(w)*, $\lambda_{sem}(qp, ne)$, *and* $\lambda_{lex}(w, v)$ can be pre-stored and easily used by IR for directly selecting answers from large sets of documents.

3. Experimental Results

Our training QA corpus consists of 2 million FAQ-Answer pairs mined from 80,000 publicly available *.com* and *.org* websites. We evaluated our approach on two different QA tasks: TREC 2003 QA Track – the Passage task, and a FAQ answer finding task. The TREC QA Passage task tests a system's ability to find an answer to a factoid question with a relatively short (250 characters) span of text [3]. A factoid question is a question asking for simple facts or relations such as *"How high is the pitcher's mound?"*. The document collection used as the source of answers consists of approximately 1,033,000 newswire documents and 3 gigabytes of text. The test question set contains 413 questions drawn from AOL and MSN Search logs. Thirty of these questions have no known correct answer in the document collection. Our experiments excluded these thirty questions from the test. When constructing the QA system for this task, we split each source document into short paragraphs (less than 250 characters), stemmed each of them, and labeled each paragraph with NE tags and Part-of-Speech tags [2]. QA performances with this data are given in Table 1. The IR-based baseline achieved 19.0% answer accuracy. We used Lucene [13] as the IR engine. Answer accuracy increased to 20.1% using the QA match score given in (7), where only the Feature Set I was considered and parameters *qf(w)* were learned from our training data using the perceptron algorithm. This performance improved further to 28.6% when using parameters learned for Feature-I and Feature-II. The QA match score with Feature-I and Feature-II was defined in formula (8). When exploiting all three types of features as given in formula (9), the answer accuracy increased to 35.7% using the parameters learned from the perceptron training. The overall absolute accuracy improvement is 16.7%.

Table 1: Experimental results: answer accuracy

Approaches		Answer Accuracy	
		TREC QA	Answer-Finding
IR		19.0%	43.0%
Perceptron Training	Feature-I	20.1%	47.5%
	Feature-I, II	28.6%	49.1%
	Feature-I, II, III	35.7%	55.5%
Absolute Improvement		16.7%	12.5%

In the FAQ answer-finding task, we collected 122,363 FAQ-answer pairs as our test data, which were also mined from the Web but not included in the training data. We used the 122,363 questions as the test questions and used the collection of 122,363 answers as the source from where answers would be chosen. Similar to the way we processed the TREC documents, we preprocessed these answers with stemming, NE tagging, and Part-of-Speech Tagging. Since these answers were prepackaged, we referred to this task as answer-finding. The answer-finding accuracies with this data are given in the last column of Table-1. The overall accuracy improvement with all three types of features is 12.5%. Different from the TREC QA task, over 60% of the questions in this data set are non-factoid questions such as "why" and "how to compare". This partially explains why Feature-I resulted more significant contribution for this data set, while Feature-II didn't improve the performance as much as it did on the TREC QA task.

4. Conclusion

This paper describes a discriminative learning approach for question answering based on a proposed variant of the voted perceptron training algorithm. Learning included (a) weighting query words in terms of their importance to retrieve the correct answer; (b) modeling the lexical association between questions and answers; and (c) modeling the semantic association between questions and answers. We built a training corpus consisting of 2 million FAQ and answer pairs by mining the World Wide Web. We tested our approach using two data sets: TREC 2003 QA data and a set FAQ-Answer pairs. We observed that:

- For the TREC data, we achieved 16.7% absolute improvement in answer accuracy over an IR-based baseline of 19% answer accuracy.
- For the FAQ answer finding task, we observed 12.5% absolute improvement in answer accuracy over an IR –based baseline of 43%.

5. References

[1] J. Lin, A. Fernandes, B. Katz, G.Marton, and S. Tellex, "Extracting Answers from the Web Using Knowledge Annotation and Knowledge Mining Techniques", Proceedings of the Eleventh Text REtrieval Conference (TREC 2002), November, 2002

[2] J. Feng, S. Reddy, M. Sarclar, "WebTalk: Mining Websites for Interactively Answering Questions", Proceedings. of Interspeech2005, September, 2005

[3] E. M. Voorhees, "Overview of the TREC 2003 Question Answering Track", Proceedings of TREC 2003

[4] M. T.Maybury, "New Directions in Question Answering", November 1, 2004

[5] Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, Vibu Mittal, "Bridging the Lexical Chasm: Statistical Approaches to Answer-Finding", Research and Development in Information Retrieval, pages 192-199

[6] R. Soricut, E. Brill, "Automatic Question Answering: Beyond the Factoid", Proc. of HLT-NAACL 2004

[7] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics", 19(2):263—312, 1993

[8] G. Salton and M. McGill, "Introduction to Modern Information Retrieval", McGraw-Hill, 1983.

[9] S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, J. Williams, J. Bensley, "Answer Mining by Combining Extraction Techniques with Abductive Reasoning", Proc. Of TREC 2003

[10] M. Collins, "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms", Proc. of EMNLP 2002

[11] X. Li, D. Roth, "Learning Question Classifiers", Proc. Of COLING'02, August. 2002

[12] Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morarescu, Finley Lacatusu, Adrian Novischi, Adriana Badulescu, and Orest Bolohan. "LCC tools for question answering", Proc. Of TREC 2002, November 2002.

[13] Otis Gospodnetic and Erik Hatcher, "Lucene in Action", 2004.