Bridging the Lexical Chasm: Statistical Approaches to Answer-Finding

Adam Berger Rich Caruana David Cohn Dayne Freitag Vibhu Mittal

Just Research
4616 Henry Street
Pittsburgh, PA 15213
U.S.A.

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
U.S.A.

Abstract This paper investigates whether a machine can automatically learn the task of finding, within a large collection of candidate responses, the answers to questions. The learning process consists of inspecting a collection of answered questions and characterizing the relation between question and answer with a statistical model. For the purpose of learning this relation, we propose two sources of data: Usenet FAQ documents and customer service call-center dialogues from a large retail company. We will show that the task of "answer-finding" differs from both document retrieval and traditional question-answering, presenting challenges different from those found in these problems. The central aim of this work is to discover, through theoretical and empirical investigation, those statistical techniques best suited to the answer-finding problem.

1 Introduction

Searching the web or skimming a lengthy manual to find the answer to a specific question can be a tedious exercise. Moreover, for a large retail company, employing a battalion of customer-support personnel to perform this same task on behalf of telephone customers can be an expensive proposition. A recent study has concluded that providing help to a single customer via a live telephone operator can cost a company \$20 to \$25 per call [8]. In this paper, we investigate statistical techniques for automating the process of answer-finding. The ultimate goal is a system which, equipped with a large collection of prepackaged answers, can automatically identify the best response to a user's query.

The approach we take is heavily inspired by machine learning. Starting from a large collection of answered questions, the algorithms we describe learn lexical correlations between questions and answers. For instance,

- Questions containing the word `why` are more likely, in general, to be paired with an answer beginning with the word `because`.
- A question containing the word `vacation` is likely to be paired with an answer containing one of the words

{`flight, trip, cruise`}.

Usenet FAQ	Ben & Jerry's FAQ
What is ViewKit? Is there a free version?	*Do you use only free range dairy?*
ViewKit is an enhanced version of the C++/Motif framework that ...	Hello. Well, other than putting up fences so they won't wander to East Pakistan, ...
What's the difference between strong AI and weak AI?	*Do you still sell Chubby Hubby T Shirts? If so, where can I find one? Thanks.*
Strong AI makes the bold claim that computer can be made to think on a level (at least) equal to humans …	Hello. We no longer sell Chubby Hubby T Shirts. Sorry. To see what we do sell, ...
How can I get my cisco to talk to a third-party router over a serial link?	*I have been reading about your new smoothie product for six months or so. Is it being distributed anywhere in Lincoln NE?*
You need to tell your cisco to use the same link-level protocol as the other router; by default, ciscos ...	We are sorry to tell you that our frozen smothies have no distribution in Nebraska...

Figure 1: Excerpts from two of the question/answer corpora used here. *Left*: Q/A pairs from the Usenet comp.* newsgroups. *Right*: Q/A pairs from Ben & Jerry's customer support.

The rest of this paper explores a series of algorithms for mining correlations between questions and answers automatically from a corpus of answered questions. To serve as a collection of answered questions, we have assembled two types of datasets:

Usenet FAQs: A collection of Usenet frequently-asked question (FAQ) documents. This dataset, a dynamic entity available publicly on the web[1], presently contains several thousand individual FAQ documents, totalling hundreds of megabytes. The topics of these documents range from libertarianism to livestock predators to programming in Fortran. For experimental purposes, we selected a set of 200 documents from the `comp.*` Usenet hierarchy containing 1800 questions.

Call-center dialogues: A collection of questions submitted by customers to Ben & Jerrys, along with the answer supplied by a company representative. This dataset contained 5145 question/answer pairs. When necessary, we also used data from an Air Canada call center containing 2889 question/answer pairs.

[1] The Usenet FAQ collection is available at `ftp://rtfm.mit.edu` and `http://www.faqs.org`. The Ben & Jerrys dataset is proprietary.

One can cast answer-finding as a traditional document retrieval problem by considering each answer as an isolated document and viewing the query as just another (albeit smaller) document. Traditional tfidf-based ranking of answers will reward candidate answers with many words in common with the query.

Employing traditional vector-space retrieval to find answers seems attractive, since `tf·idf` is a standard, time-tested algorithm in the toolbox of any IR professional. However, the experiments reported in this paper demonstrate that standard `tf·idf` retrieval performs poorly compared with techniques that "learn" to locate answers by inspection of a collection of answered questions.

1.1 The lexical chasm

In ranking documents by relevance to a query, traditional information retrieval systems place a large emphasis on lexical similarity between document and query: the closer the distribution of words in a candidate document is to the query, the more relevant is the question. Most users of document retrieval systems have this model in mind, and in formulating their query they usually employ terms that they expect would appear in a relevant document. But users who submit questions to an answer-finding system can't be expected to anticipate the lexical content of an optimal response: there is often very little overlap between the terms in a question and the terms appearing in its answer. For example, the best response to the question `Where's a good place to get dinner?` might be `Zaphod's Bar and Grill has great fajitas.` which have *no* tokens in common.

More generally, questions often contain terms that differ from, but are related to, the terms in the matching answer. The group of terms {`what`, `when`, `where`, `why`, `how`} will typically appear more frequently in questions than answers, for example. The legal vocabularies for questions and answers are the same, but the probability distributions over those vocabularies are different for questions and their answers.

Furthermore, the probability distribution for terms in the answer is linked to the probability distribution of the terms in the question. Thus there is both a mismatch between the terms in queries and the terms in responses matching those queries, as well as a correspondence between the mismatched terms in the query and response. For example, in a `where` question, the response frequently contains the words {`near, adjacent, street, on`} and so forth.

We refer to this combination of a vocabulary mismatch and linkage between query and response vocabularies as a *lexical chasm*. The query is on one side of the chasm and the response on the other side. The vocabularies on the two sides of the chasm are the same, but the distributions differ on each side of the chasm. The distributions on the two sides of the chasm are linked at the semantic and discourse levels.

This chasm suggests that traditional bag-of-words retrieval might be less effective at matching questions to responses than matching keywords to documents. To bridge the lexical chasm, an IR system must adopt a strategy that rises from the lexical level towards the semantic level. All the approaches we describe in this paper do this. For instance, latent semantic indexing can be interpreted as learning the linkage between the two probability distributions on opposite sides of the chasm by discovering and then exploiting intermediate concepts that link the two sides.

Traditional IR systems based on the `tf·idf` ranking criterion [11] suffer from a particular form of the lexical gap problem, namely the problem of synonymy: a query containing the term `Constantinople` ought to fetch documents about Istanbul, but doing so requires a step beyond comparing the word frequency histograms in query and candidate documents. Synonymy has received much attention within the document retrieval community recently, and researchers have applied a variety of heuristic and statistical techniques—including pseudo-relevance feedback, local context analysis, and probabilistic models of synonymy [2, 7, 13]. We anticipate that although document retrieval and answer-finding are distinct problems, there will be cross-fertilization between these areas. In fact, in Section 3 we apply adapted versions of query expansion and statistical translation to the answer-finding problem.

1.2 Strategies for answer-finding

The first learning technique we discuss, ADAPTIVE TFIDF, is an extension of the `tf·idf` algorithm. The idea is to adjust the idf-weights of each word so as to maximize retrieval of the correct answer for each question in the training set. Adaptive `tf·idf` doesn't represent an attempt to bridge the aforementioned lexical gap, but merely to exploit labeled training data to improve the baseline `tf·idf` performance.

AUTOMATIC QUERY EXPANSION, a simple yet effective technique in traditional document retrieval, is a strategy for refining a query by adding terms to it. In Section 2 we introduce a technique for learning, from a collection of answered questions, which answer words (like `circa` and `yesterday`) are most strongly correlated with a question-word (like `when`). Equipped with this information, the system can then automatically add these answer words to "help" a query containing that word.

The next technique we introduce, STATISTICAL TRANSLATION MODELS, explicitly attempt to bridge the lexical gap between questions and answers by characterizing the co-occurrence between one word in a query and another word in an answer. The idea is to calculate the entries of a two-dimensional stochastic matrix, where the i, jth cell reflects the likelihood that an answer containing the word j corresponds to a question containing word i.

LATENT VARIABLE MODELS may be thought of in this context as a strategy for performing soft clustering of questions and answers. The clustering is "soft" in the sense that each question and answer belongs, to a varying degree, to each cluster; the clusters are akin to topics.

Table 1 summarizes the techniques we investigate in this paper.

1.3 Evaluation Criteria

This paper empirically evaluates several approaches to answer finding. Each method is compared to an appropriate "baseline" method: (1) the baseline for adaptive `tf·idf` is the normal `tf·idf` (2) the baseline for query-expansion is the original query without expansion, (3) for statistical translation, the baseline is the performance of bayesian retrieval with parameters learned from just the individual answers, and (4) for the latent variable models, the baseline is `tf·idf`.

tfidf	Traditional vector-space approach: how closely do the term frequencies in the question and candidate answer match?
adaptive tfidf	tfidf + machine learning: adjust term weights to optimize performance on training Q/A pairs
query expansion	Learn which answer terms appear with which question terms by inspecting training data, in order to pad new questions
statistical translation	From training Q/A pairs, construct a statistical model for how a query "translates" to its answer.
latent variable models	Each question and answer arises from a mix of (unseen) topics; characterize these topics by inspection of training Q/A pairs.

Table 1: Five statistical techniques to answer-finding.

All four methods are compared to the appropriate baseline on two FAQs: the Usenet comp.* dataset and the Ben & Jerry call-center dataset. In addition, adaptive `tf·idf` and latent variable models are also tested on the Air Canada call-center dataset. Each technique was evaluated by using five runs of randomly selected test sets containing ten percent of the total document set. It should be noted that the datasets we had access to contained just one question for each answer in the FAQs. This prevented us from evaluating the system where was training was conducted on a subset of the questions available for each answer and testing was done on the held-out questions; instead, we were forced to train on one set of question-answer pairs, and test on another, completely unseen set of question-answer combinations.

thus all our experiments were conducted using the question words as a "query" on the answers. Given a suitable dataset with multiple questions for an answer, it would be interesting to evaluate whether using something like `tf·idf` on the questions (or a combination of the questions and the answers) instead of the answers alone would yield any improvement in performance.

Normal precision-recall measures are less useful in evaluating question answering systems than other measures. This is because users are unlikely to be happy if the the first hundred "hits" are wrong. What counts is how close a correct answer is to the top of the returned list. Instead of precision-recall measures, we use the rank of the correct answer in the returned list as a metric of performance. We calculate two measures from the rank: the median rank of the correct answer, and the inverse harmonic mean rank. The advantage of the median is that it is less affected by non-representative tails of the distribution. The inverse harmonic mean rank (computed by inverting the average of the inverse ranks of the correct answers) is very good at giving an intuitive feel for where the correct answer is likely to appear. It also penalizes rank changes near the top more than changes farther away (thus a drop in rank from 2 to 3 is much more significant than a change from 99 to 100 and is reflected in this measure).

The paper proceed as follows. The next section describes the statistical learning approaches we applied to the answer-finding problem. Since there is much to answer-finding *not* described in this paper, Section 4 discusses some promising extensions of this work. Section 5 describes the work in question-answering and document retrieval which we feel is most closely related to ours.

2 Statistical models for answer-finding

This section discusses several increasingly sophisticated techniques for building and using an answer-finding system. We begin with `tf·idf` as a baseline. Two of the following four methods build on `tf·idf`. The methods, however, are equally applicable to many other retrieval algorithms. Each section presents empirical results comparing that method to the appropriate baseline method.

2.1 `tf·idf`

Given an m-word query $\mathbf{q} = \{q_1, q_2, \ldots q_m\}$, an n-word answer $\mathbf{a} = \{a_1, a_2, \ldots a_n\}$ and a set of N recognized words, one can represent $\mathbf{q}$ and $\mathbf{a}$ each as a vector of word frequencies $\vec{\mathbf{q}}$ and $\vec{\mathbf{a}}$. By $\vec{\mathbf{a}}_k$ we mean the frequency with which the kth word appears in $\mathbf{a}$.

A common measure of the similarity between two word frequency vectors $\vec{\mathbf{a}}$ and $\vec{\mathbf{q}}$ is the cosine distance between them:

$$\begin{aligned} score(\mathbf{q}, \mathbf{a}) &\stackrel{\mathrm{def}}{=} \frac{\vec{\mathbf{q}} \cdot \vec{\mathbf{a}}}{||\vec{\mathbf{q}}|| \cdot ||\vec{\mathbf{a}}||} \\ &= \frac{\sum_{w \in \mathbf{q},\mathbf{a}} f_{\mathbf{q}}(w) \cdot f_{\mathbf{a}}(w)}{\sqrt{\sum_{w \in \mathbf{q}} f_{\mathbf{q}}(w)^2 \cdot \sum_{w \in \mathbf{a}} f_{\mathbf{a}}(w)^2}}, \end{aligned} \tag{1}$$

where $f_{\mathbf{d}}(w)$ is the number of times word w appears in document $\mathbf{d}$. This scoring function weighs each word equally, even though knowing the frequency of a word like `fractal` in a document tells us much more about the document than knowing the frequency of words like `and`. Inverse document frequency (`idf`) weighting is a popular IR method for weighting terms by their "information content," taken to be related to the frequency with which documents contain that term [11]. In our case, a document is an answer; we denote the entire set of answers by $\mathcal{D}$, and weight each term w according to

$$\lambda_w \stackrel{\mathrm{def}}{=} \mathtt{idf}(w) = \log\left(\frac{|\ \mathcal{D}\ |}{|\ \{\mathbf{d} \in \mathcal{D} : f_{\mathbf{d}}(w) > 0\}\ |}\right). \tag{2}$$

Given these weights, we compute a weighted score[2]

$$score(\mathbf{q}, \mathbf{a}) = \frac{\sum_{w \in \mathbf{q},\mathbf{a}} \lambda_w^2 \cdot f_{\mathbf{q}}(w) \cdot f_{\mathbf{a}}(w)}{\sqrt{\sum_{w \in \mathbf{q}} f_{\mathbf{q}}(w)^2 \cdot \sum_{w \in \mathbf{a}} f_{\mathbf{a}}(w)^2}}. \tag{3}$$

Baseline results for traditional `tf·idf` are included below where appropriate.

2.2 Adaptive `tf·idf`

`tf·idf` performs reasonably well despite its simplicity. Given a training set of question-answer pairs, however, there is room for improvement. One approach is to do some form of hill-climbing for each w to bring a question and its corresponding answer "closer"—raise $score(\mathbf{q}, \mathbf{a})$, that is. An intuitive and

[2] In addition to variations on computation of `idf`weights, there are variations on how they are incorporated into the cosine distance. The formulation given here is the one that gave the best *ab initio* median rank retrieval before learning was applied.

Usenet `comp.*` FAQs

Method	Median	p	Harmonic Mean	p
baseline	2	-	1.90	-
adaptive	2	-	1.90	0.878

Air Canada Call Center FAQ

Method	Median	p	Harmonic Mean	p
baseline	10.0	-	3.60	-
adaptive	7.4	0.003	3.22	0.005

Ben & Jerry's Call Center FAQ

Method	Median	p	Harmonic Mean	p
baseline	16.6	-	3.91	-
adaptive	13.0	0.033	3.69	0.003

Table 2: Experiments with adaptive `tf·idf` on the Usenet, and two call-center datasets. The numbers here are averaged over 5 runs of randomly selected testing set of 10% of the document sets. p values are paired t-statistics for the test that the adaptive `tf·idf` outperforms the baseline.

effective form of hillclimbing is to do gradient descent on λ. The gradient of (3) with respect to term weight is

$$\frac{\partial score(\mathbf{q},\mathbf{a})}{\partial \lambda_w} = \left(\frac{2}{\lambda_w}\right) score(\mathbf{q},\mathbf{a}) \tag{4}$$

We can bring $\mathbf{q}$ and the correct answer $\mathbf{a}^\star$ closer together—raise $score(\mathbf{q},\mathbf{a})$, that is—by iteratively setting

$$\lambda_w \leftarrow \lambda_w + \gamma\left(\frac{2}{\lambda_w}\right) score(q,a) \text{ for all } w,$$

where $0 < \gamma \leq 1$ is a learning rate parameter.

Unfortunately, merely bringing the question and answer closer together is not sufficient, for the weight update may also bring other, incorrect answers closer to the question. What we *really* want to do is hillclimb on the retrieved rank of the correct answer. Rank is a discrete combinatorial quantity, so there is no easy way to hillclimb on rank directly. Instead, we apply the following heuristic: for all question/answer pairs in the training set for which the system did not rank the correct answer first, hillclimb *up* on the the correct answer and hillclimb *down* on all incorrect answers scoring better than the correct one:

$$\begin{aligned} \lambda_w &\leftarrow \lambda_w + \gamma\left(\frac{2}{\lambda_w}\right) score(\mathbf{q},\mathbf{a}^\star) \\ \lambda_w &\leftarrow \lambda_w - \gamma\left(\frac{2}{\lambda_w}\right) score(\mathbf{q},\mathbf{a}') \end{aligned}$$

where the second update is applied for all $\mathbf{a}'$ satisfying $score(\mathbf{q},\mathbf{a}') \geq score(\mathbf{q},\mathbf{a}^\star)$.

We hold out a portion of the training set for validation, and hillclimb until performance on that set stops improving. This idea—of adjusting parameters to increase the score of the correct answer while decreasing the score of incorrect answers—is somewhat related to the emerging technique of discriminative training.

Table 2 presents the results of a 5-fold experiment on three datasets. The rows labeled "baseline" show the performance of unadjusted `tf·idf`, while "adaptive" represents `tf·idf` with learned `idf` factors. While we see some improvements on all datasets, the greatest improvements are realized on the call center data. We speculate that the greater size and homogeneity of these datasets provide more opportunity for improvement by adjusting `idf` factors.

2.3 Query expansion

With this strategy we finally begin to attempt to bridge the lexical chasm between questions and answers. In the context of document retrieval, traditional query expansion typically involves adding words to the query which are likely synonyms of (or at least related to) words in the original query. When we have access to a training set of paired questions and responses, however, we have the opportunity for a more focused kind of expansion: learn a mapping between query terms and their counterparts in the responses (like `why` $\rightarrow$ `because`, `site` $\rightarrow$ `http`, and `windows` $\rightarrow$ `Microsoft`) and add those terms to the query that the mapping suggests will appear in the answer.

We learn this mapping by calculating the mutual information between query terms and answer terms in the training set:

$$\begin{aligned} I(u,v) &= H(p(v \in \mathbf{a})) \\ &\quad -p(u \in \mathbf{q})H(p(v \in \mathbf{a} \mid u \in \mathbf{q})) \\ &\quad -p(u \notin \mathbf{q})H(p(v \in \mathbf{a} \mid u \notin \mathbf{q})) \end{aligned}$$

where $w \in \mathbf{q}$ is shorthand for a binary random value:

$$w \in \mathbf{q} = \begin{cases} 1 & \text{if word } w \text{ appears in } \mathbf{q} \\ 0 & \text{otherwise,} \end{cases}$$

and $p(u \in \mathbf{a} \mid v \in \mathbf{q})$ denotes the conditional probability of u appearing in an answer if v appears in the corresponding query. The function $H(\cdot)$ is the entropy:

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

Using this formula, we can determine the top n terms in the answer terms that are correlated with any question term. These can then be used to *expand* a query in the hope that adding the additional terms will result in better performance against a testing set. To expand a query means to augment the query with the k words having the highest mutual information to each term in the query.

The results reported in Table 3, show that adding "synonymous" terms clearly helps up to a point. Even adding a single term per query term manages to improve effective performance on the *harmonic mean rank* metric by just over 50%. For the Usenet data the median is not affected, while for the call-center data set, the median is significantly reduced in the case of one and two synonyms. The mean rank does increase in both cases, reflecting the fact that in some cases, adding additional terms to the query adds more noise than signal to the query. The uniform improvements in performance in both cases are significant given the wide variation between the two datasets–not just in terms of size, but also in terms of mean query length, answer length and range of topics covered.

2.4 Statistical translation

The idea that a computer could automatically learn to translate text by inspection of a large quantity of bilingual text computer was first contemplated by Warren Weaver a half century

Usenet comp.* FAQs

syn	Median	p	Harmonic Mean	p
0	3	-	4.17	-
1	3	-	2.08	<0.001
2	3	-	2.08	<0.001
3	3	-	2.09	<0.001
4	3.2	0.374	2.10	<0.001

Ben & Jerry's Call Center FAQ

syn	Median	p	Harmonic Mean	p
0	17.25	-	6.35	-
1	11.20	0.002	3.11	<0.001
2	14.60	0.075	3.25	<0.001
3	17.20	0.468	3.38	<0.001
4	20.00	-	3.49	<0.001

Table 3: Experiments with query expansion in tf·idf on the Usenet and the call-center datasets. Performance is shown on two different metrics: the median rank, and the harmonic mean rank (the inverse of the mean inverse rank). The numbers here are averaged over 5 runs of randomly selected testing set of 10% of the document sets. p values are paired t-statistics for the test that the expanded queries outperform the baseline.

Usenet comp.* FAQs

Method	Median	p	Harmonic Mean	p
baseline	3.00	-	4.12	-
translation	1.60	0.008	1.72	<0.001

Ben & Jerry's Call Center FAQ

Method	Median	p	Harmonic Mean	p
baseline	16.6	-	6.25	-
translation	25.2	-	3.41	<0.001

Table 4: Experiments with statistical translation on the Usenet, and a call-center dataset. The numbers here are averaged over 5 runs of randomly selected testing set of 10% of the document sets. p values are unpaired t-statistics for the test that the translation model outperforms the baseline.

ago [12]. Accurate automated translation of free text still appears to be years away, but the algorithms developed for statistical machine translation (MT) have recently been applied with promising results to the problem of document retrieval and summarization [2]. Here we describe how one can apply these same algorithms to the problem of question-answering.

By "translation model," we mean a conditional probability distribution $p(\mathbf{t} \mid \mathbf{s})$ over sequences of words $\mathbf{t} = \{t_1, t_2 \ldots, t_m\}$ in one language, given a sequence of words $\mathbf{s} = \{s_1, s_2, \ldots, s_n\}$ in another language. The value $p(\mathbf{t} \mid \mathbf{s})$ is the probability that, when presented with the source sequence $\mathbf{s}$, an expert translator will translate it to the target $\mathbf{t}$. A reasonable French-English translation model p, for instance, should have the property that

$p($`Le président Lincoln a été un bon avocat` $\mid$ `President Lincoln was a good lawyer`$)$

is much higher than

$p($`Je me brossé les dents` $\mid$ `President Lincoln was a good lawyer`$)$.

We apply the MT framework to question-answering as follows. The notion of "source" language corresponds to the answers, and the "target" language corresponds to the questions. As with the LM approach, we then equate the relevance of an answer $\mathbf{a}$ to a question $\mathbf{q}$ with the quantity $p(\mathbf{q} \mid \mathbf{a})$.

There are reasons to think the translation approach might work well for question-answering: trained on a sufficient amount of question/answer pairs, the translation model should learn how answer-words "translate to" question-words, bridging the lexical chasm. For instance, words like `at, location, place, street, directions` will all translate with reasonably high probability to the question-word `where`.

The translation model we use in this work is derived from the IBM family of translation models [3]. These models rely on an *alignment*, which we denote by α, between sequences of words, capturing how subsets of answer words conspire to produce each question word. Using α, we can decompose $p(\mathbf{q} \mid \mathbf{a})$ as

$$p(\mathbf{q} \mid \mathbf{a}) = \sum_{\alpha} p(\mathbf{q}, \alpha \mid \mathbf{a}) = \sum_{\alpha} p(\mathbf{q} \mid \alpha, \mathbf{a}) p(\alpha \mid \mathbf{a}) \quad (5)$$

Assuming that exactly one source word is responsible for a given target word, we can write

$$p(\mathbf{q} \mid \alpha, \mathbf{a}) = \prod_{i=1}^{m} \tau(t_i \mid s_{a_i}) \quad (6)$$

Here s_{a_i} is the answer word aligned with the ith query word, and $\tau(q \mid a)$ is a parameter of the model—the probability that the answer word a is paired with the question word q in the alignment.

If $\mathbf{q}$ contains m words and $\mathbf{a}$ contains n words, there are n^m alignments between $\mathbf{a}$ and $\mathbf{q}$. Assuming all these alignments are *a priori* equally likely, we can write

$$p(\mathbf{q} \mid \mathbf{a}) = \frac{p(m \mid \mathbf{a})}{n^m} \sum_{a} m \prod_{i=1}^{m} \tau(t_i \mid s_{a_i}) \quad (7)$$

Given a collection of bilingual sentences

$$\mathcal{C} = \{(\mathbf{q}_1, \mathbf{a}_1), (\mathbf{q}_2, \mathbf{a}_2), (\mathbf{q}_3, \mathbf{a}_3) \ldots,$$

the likelihood method suggests that one should adjust the parameters of (7) in such a way that the model assigns as high a probability as possible to $\mathcal{C}$. This maximization must be performed, of course, subject to the constraints $\sum_q \tau(q \mid a) = 1$ for all terms a. One can use the EM algorithm for this purpose, as developed by Brown *et al* [3].

Having learned the word-to-word synonymy parameters from the training data, a translation-based system is then ready to perform answer-finding as follows. Starting from an input question $\mathbf{q}$, rank each answer according to $p(\mathbf{q} \mid \mathbf{a})$ via (7). The sum in (7) is over an exponential number of alignments, but one can calculate this value efficiently by rearranging the sum and product.

Space limitations prohibit a detailed explanation of the training or ranking mechanism, but the interested reader may fill in the details by consulting the references [3].

As Table 4 indicates, statistical translation can lead to significant ranking improvements. The only exception is the median rank on the Ben & Jerry's problem. Interestingly, while the median rank falls, the harmonic mean rises considerably. The harmonic mean is more sensitive to smaller numbers (documents with higher rank). This suggests a higher fraction of correct documents ranked close to the top than with `tf·idf`—the behavior we expect from an answer-finding system.

2.5 Latent Variable Models

We can combine several ideas from the previous two sections to form yet another approach to the problem. Statistical translation effectively builds a full joint p.d.f. to probabilistically "translate" words in the question to words that should appear in the answer. Aside from the data and memory requirements of building the full p.d.f., there may be other advantages to finding a "compressed" representation. Latent variable models such as LSA [5] and PLSI [9] have demonstrated that factored representations can capture some semantic information, by forcing the model to represent words used in similar contexts to be mapped to each other.

One factored model of translation is

$$p(w_a|w_q) = \sum_z p(w_a|z)p(z|w_q)$$

, where z is a vector of factors that mix to produce an observation. The quantities $p(w_a|z)$ and $p(z|w_q)$ are found probabilistically, by applying the EM algorithm [6] to observed question-answer document pairs. Empirically, the derived factors frequently correspond to "topics" or "aspects" covered by a document, with semantically similar terms covered by a common factor.

Hofmann [9] demonstrates how PLSI can be mixed with `tf·idf` to improve retrieval performance on standard IR problems. His model however, uses the standard two-way, term-document factoring, translating answer terms to other answer terms; it is unable to represent the two different types of documents (questions and answers) found in FAQs, or their distinct word distributions. Explicitly distinguishing these documents and their vocabularies, and linking them via common latent factors yields a four-way factoring (see Figure 2).

The latent variable model may be justified by a factored generative model of question answering. With some probability $p(\mathbf{z})$, the user is interested in some topic, represented by the mixture of factors $\mathbf{z}$. Based on this interest she constructs a question document $\mathbf{q}$, drawing her words from a topic-specific distribution of question words, $p(w_q|\mathbf{z})$. We assume that for each topic, or mix thereof, there is a distribution of answer documents $p(\mathbf{a}|\mathbf{z})$, with vocabularies drawn from an independent, but (probably) overlapping word distribution $p(w_a|z)$. Our task is to find the answer document most closely matching the topic that evoked the question: $\arg\max_a p(\mathbf{a}|\mathbf{q})$.[3]

Empirically, simply maximizing this quantity does not perform well for practical sizes of z. Instead, we follow Hofmann's approach, and mix the factored model with `tf·idf` as follows: We begin by finding the most likely factor mix for the question

[3] Notice that one again the order of prediction is in the "forward" direction: predicting the answer from the query. The two directions are in fact equivalent for the purposes of scoring answers if one assumes a uniform prior over answers.

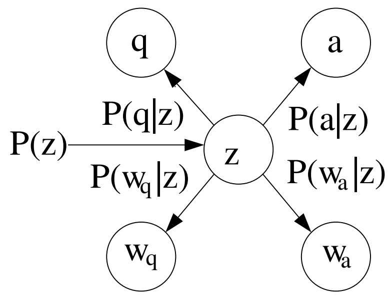

Figure 2: Four-way factored model of FAQ generation.

document—computing

$$\arg\max_z \prod_{w_q \in q} \sum_z p(w_q|z)p(z). \tag{8}$$

Given z, we know $p(w_a|z)$ is the distribution of words we expect to see in the corresponding answer a. We generate a pseudo-answer a' as the weighted vector of words for which $p(w_a|z)$ is non-negligible, and perform `tf·idf` retrieval of a using a' as the query. In practice, we have observed best performance when a' and the original question q are mixed, weighting terms in a' by α and terms in q by $1-\alpha$.

To build the factored model, we begin with a set of questions documents and corresponding answer documents, and search for a set of parameters $p(z)$, $p(q|z)$, $p(a|z)$, $p(w_q|z)$ and $p(w_a|z)$ which maximize the likelihood of the training data. The remaining quantities, such as $p(z|q)$, can then be computed with Bayes rule.

Maximizing likelihood is done by applying a form of the EM algorithm. We begin with an arbitrary assignment of likelihoods to $p(q|z)$, $p(a|z)$, $p(w_q|z)$ and $p(w_a|z)$ and estimate the unknown topic mixture as

$$p(z|q,a,w_q,w_a) = \frac{p(q|z)p(a|z)p(w_q|z)p(w_a|z)p(z)}{\sum_{z'} p(q|z')p(a|z')p(w_q|z')p(w_a|z')p(z')}. \tag{9}$$

From the topic mixture estimates, we recompute the likelihood of the data for a given mixture. We define $n(q, w_q)$ as the number of times word w_q appears in document q. Then, for all matching pairs of documents q and a we calculate:

$$p(q|z) = \sum_{w_q,a,w_a} \vartheta(q,a,w_q,w_a)/Z \tag{10}$$

$$p(a|z) = \sum_{q,w_q,w_a} \vartheta(q,a,w_q,w_a)/Z \tag{11}$$

$$p(w_q|z) = \sum_{q,a,w_a} \vartheta(q,a,w_q,w_a)/Z \tag{12}$$

$$p(w_a|z) = \sum_{q,d_a,w_q} \vartheta(q,a,w_q,w_a)/Z \tag{13}$$

where

$$\begin{aligned} \vartheta(q,a,w_q,w_a) &\stackrel{\text{def}}{=} n(q,w_q)\cdot n(a,w_a)\cdot p(z|q,a,w_q,w_a) \\ Z &\stackrel{\text{def}}{=} \sum_{q,w_q,a,w_a} \vartheta(q,a,w_q,w_a). \end{aligned}$$

Usenet comp.* FAQs

Method	Median	p	Harmonic Mean	p
baseline	2.00	-	1.86	-
latent	1.00	<0.001	1.30	<0.001

Air Canada Call Center FAQ

Method	Median	p	Harmonic Mean	p
baseline	10.8	-	3.65	-
latent	4.80	0.003	2.42	<0.001

Ben & Jerry's Call Center FAQ

Method	Median	p	Harmonic Mean	p
baseline	16.0	-	3.88	-
latent	6.40	<0.001	2.67	<0.001

Table 5: Experiments with a latent variable on the Usenet, and two call-center datasets. The numbers here are averaged over 5 runs of randomly selected testing set of 10% of the document sets. p values are paired t-statistics for the test that the latent variable model outperforms the baseline.

By iteratively applying (9) and (10)–(13), we are guaranteed to converge to a locally optimal generative model of the data.

As shown in Table 5, latent variable models improve on `tf·idf` uniformly by significant margins. The harmonic mean ranks achieved by this method are consistently low, pointing to its ability to rank a large fraction of the correct answers at or near the top.

3 Discussion of experimental results

All of the techniques we present outperform `tf·idf` on the task of identifying the correct answer to a question from among a large pool of candidate answers. The techniques we have explored are diverse. What unifies them is the idea of exploiting the availability of a training corpus of question/answer pairs.

It appears that even rather unsophisticated methods such as adjusting `idf` factors or learning how to add answer terms to queries can enhance the performance of an answer-finding system. In order to make best use of the training corpus, however, our experiments suggest the use of more sophisticated probabilistic models.

4 Extensions

It could be argued (as it has for the related problem of question-answering) that essentially linguistic-free approaches such as those described here are inherently limited: to really solve the answer-finding problem may involve a deep linguistic or even semantic analysis of the question. However, even within a purely lexical, statistical framework, there is much room for improvement. This section contains some of what we consider to be "low-hanging fruit" in the domain of answer-finding.

4.1 Using the questions

The answer-finding approaches we investigated all view FAQ data as a collection of answers to be ranked. Of course, to each answer in a FAQ document corresponds a question. A real-world answer-retrieval system would exploit the questions as well as the answers when searching for the most appropriate answer to a new question. For instance, if a user asks a question nearly identical to one already appearing in the database, the system clearly ought to recognize this and respond with the corresponding answer.

Pursuing this line of investigation here would have been difficult with our datasets. We would like to learn how a single question can have multiple representations, but our data do not contain multiple synonymous question. If more than one question were available for each answer, we could learn how to map a question onto a question/answer pair, rather than just onto an answer. Future research in FAQ answer finding would benefit tremendously from FAQ databases that keep track of alternate user questions that get mapped to each reply in the FAQ, and we are exploring ways to construct such a dataset.

4.2 Exploiting document structure

The experiments reported in this paper treat the question/answer pairs as isolated objects. In fact, they often occur as part of a larger document structure. There are several strategies for exploiting this structure to improve the accuracy of answer retrieval. One idea is to try to find not only the individual answer best matching the input question, but to find the best *region*—collection of answers, say. Giving a user a larger body of text which probably contains the correct answer is worse than providing just the answer, but better than providing the wrong region, or no region at all.

Another approach is to introduce a function of the position of an answer in an FAQ as a prior probability that the answer is appropriate. It may be, for example, that simpler, more general questions usually occur early in a user's manual, and people generally ask more general questions first; an obvious strategy in this case would be to bias towards the first several answers early in a dialogue with a user.

4.3 Exploiting question structure

The answers in an FAQ are more than just regular documents; they are answers to questions. Different types of questions lead to different types of answers. The correspondence between question types and answer types might be exploited to improve retrieval accuracy.

Suppose for the sake of argument that there are exactly five different kinds of questions: `what`, `when`, `where`, `why`, and `how`. Further suppose that FAQ questions (and user queries) are evenly divided, i.e., that 20% of queries are `what` queries, etc. Finally, assume that the type of a query can be identified with perfect accuracy, and that the type of an answer can also be identified with perfect accuracy. If we identify the type of the user's query as a `why`-type, and can identify which answers in the FAQ are answers to question of type `why`, then we need only search 20% of the answers for the best match to the user's query. This could lead to a dramatic improvement in the accuracy of the retrieved answers. Unfortunately, the problem of identifying question type has long been recognized as a difficult one. [10].

Is the question "How do I get to the World Trade Center" really a `how`-question, or could it be considered a `Where`-question?

5 Related work

This paper focuses on the task of locating an answer within a large collection of candidate answers. This is to be contrasted with the problem of *question-answering*, a considerably more ambitious endeavor [1] requiring the construction of an answer by searching a large collection of text. Question answering systems are usually domain-specific and highly knowledge-intensive, applying sophisticated linguistic analysis to both the question and the text to be searched for an answer.

Somewhat more closely related to this work is the FAQ-FINDER system under development at U.C. Irvine [4]. The system attempts to locate, within a collection of Usenet FAQ documents, the most appropriate answer to an input question. The FAQ-FINDER system is similar to the work described in this paper in starting with a `tf·idf`-based answer-scoring approach. In trying to bridge the lexical chasm, however, our paths diverge: FAQ-FINDER relies on a semantic network to establish correlations between related terms such as `husband` and `spouse`. In contrast, our approaches depend only on the availability of a suitable training set, and do not require external knowledge sources such as wide-coverage semantic network. Moreover, the machine learning approaches we discuss can adapt themselves to changes in terminology and usage patterns.

6 Conclusions

This paper presents an empirical study on four statistical techniques for *answer-finding*, an emerging problem in IR related to both document retrieval and question-answering. To allow for a quantitative analysis of the candidate algorithms, we employed two real-world datasets: a collection of Usenet FAQs and a set of customer-response dialogues. Each of the four techniques performed quite well, and depending on the characteristics of the underlying data set –vocabulary size, the overlap between questions and answers, and between multiple answers, etc. – and the desired performance level (and computational cost) may be best in different situations. In our experiments here, the more highly parametrized techniques based on statistical translation and aspect models seemed to be better at bridging the "lexical chasm" between questions and answers than the `tf·idf` based ones.

Though the work described here in this paper did not incorporate any linguistic processing, we believe that in many cases there can be potential advantages to using either linguistic or semantic analyses of the question and answers; it is likely that building a better answer-finding system may require a combination of both statistical and linguistic approaches. Due to reasons beyond our control, we were unable to discuss in this paper the effectiveness of combining the four statistical methods presented here. One way to combine multiple methods – such as the four discussed in this paper – is to use a weighted combinination of the ranks produced by the individual methods. The appropriate weights could be learned from a training set, enabling an answer finding system to be optimized for a particular application or domain.

Acknowledgments

This research was supported in part by an IBM University Partnership Award and by Claritech Corporation. The authors thank RightNow Technologies for access to the call-center data used in Section 3.

References

[1] AAAI. *Proceedings of the AAAI FSS on Question Answering Systems* (Cape Cod, MA, November 1999).

[2] Berger, A., and Lafferty, J. Information retrieval as statistical translation. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*. Berkeley, CA, 1999.

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

[4] Burke, R., Hammond, K., Kulyukin, V., Lytinen, S., and Tomuro, N. Question answering from frequently-asked question files: Experiences with the FAQ Finder system. Tech. Rep. TR-97-05, Department of Computer Science, University of Chicago, 1997.

[5] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T., and Harshman, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, 6 (1990), 391–407.

[6] Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society 39* (1977), 1–38.

[7] Efthimiadis, E., and Biron, P. UCLA-Okapi at TREC-2: Query expansion experiments. In *Proceedings of the Second Text Retrieval Conference* (1994).

[8] GARTNER GROUP. Gartner group report, 1998.

[9] Hofmann, T. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval* (1999).

[10] Lehnert, W. *The process of question answering: A computer simulation of cognition*. Lawrence Erlbaum Associates, 1978.

[11] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management 24* (1988), 513–523.

[12] Weaver, W. Translation (1949). In *Machine Translation of Languages*. MIT Press, 1955.

[13] Xu, J., and Croft, B. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual ACM Conference on Research and Development in Information Retrieval*. 1996.