Discovering links between lexical and surface features in questions and answers

Soumen Chakrabarti[*]
IIT Bombay
soumen@cse.iitb.ac.in

ABSTRACT

Information retrieval systems, based on keyword match, are evolving to question answering systems that return short passages or direct answers to questions, rather than URLs pointing to whole pages. Most open-domain question answering systems depend on elaborately designed hierarchies of question types. A question is first classified to a fixed type, and then hand-engineered rules associated with the type yield keywords and/or predictive annotations that are likely to match indexed answer passages. Here we seek a more data-driven approach, assisted by machine learning. We propose a simple conditional exponential model over a pair of feature vectors, one derived from the question and the other derived from the a candidate passage. Features are extracted using a lexical network and surface context as in named entity extraction, except that there is no direct supervision available in the form of fixed entity types and their examples. Using the exponential model, we filter candidate passages and see substantial improvement in the mean rank at which the first answer is found. The model parameters distill and reveal linguistic artifacts coupling questions and their answers, which can be used for better annotation and indexing.

1. INTRODUCTION

A Question Answering (QA) system responds to queries like *Who is the Greek God of the Sea?* with a precise answer like *Poseidon*. An important first step, which we focus on here, is to identify short *snippets* or passages of up to several words which contain the answer.

Traditionally, Web search engines have not rewarded queries that are grammatical and coherent. Users have adapted by translating their information need (e.g., *When was the Space Needle built?*) into telegraphic keyword queries (e.g., `"Space Needle" history`). However, this process remains a tentative one.

Meanwhile, the QA community, building on top of Information Retrieval (IR) systems, part-of-speech and named-entity (NE, e.g., people, place, time) taggers, and shallow parsers, has made substantial advances in building high-accuracy QA systems [3, 8, 9, 11, 5, 18, 26, 6, 16].

However, the success comes at a price: significant human effort is needed to study the corpus and typical questions and their answers, and turn that experience into elaborate pattern matching and extraction strategies. The best QA systems incorporate expert input based on manual analysis of large corpora and a large collection of questions (over 27600 in one case).

Expert input is coded in at least two forms. The first is a question *typology* with several hundred question types, such as location (Where is John Wayne airport?), events (When is Bastille Day?), creation-time (When did World War 1 start?), population (What is the population of Venezuela?), why-famous (Who was Galileo?) and so on. Further wisdom is embedded in routines that produce the final ranking of passages short-listed by a conventional IR engine. The best-performing ranking routines exploit well-tuned heuristic that set rewards, discounts, and penalties for exact matches, case-insensitive matches and stemmed matches between words.

Our goal: We wish to harness robust data mining and machine learning tools to ease—and ideally automate—the process of mapping ad-hoc questions to likely answer passages, without having to fashion and maintain a question taxonomy, or to tune passage ranking heuristics through manual intervention.

We draw on two kinds of resources. First, we use a database of **is-a** relations (e.g., a horse is a mammal). WordNet[1] is a source of is-a relations widely used in natural language analysis. Moreover, recent research has enabled the bootstrapping of is-a databases, largely automatically, from the Web [7]. Many focused domains have already created ontologies suited to their use, e.g., the MeSH hierarchy for medical abstracts and the UNSPSC taxonomy for e-commerce catalogs. These knowledge bases feed naturally into our system.

Second, we borrow basic feature extraction strategies from the named entity (NE) extraction and information extraction (IE) communities. Unlike IE, we do not have a fixed set of labels. Ground features on both the question and answer sides must be abstracted into "soft labels" which allow for probabilistic matching. Intuitively, if a test question is "similar" to a sufficient number of training questions, we should be able to predict the kind of features we wish to see in an answer passage directly, without first embedding the question into a type system.

Consider questions that are answered with the name of a place. There are several ways to ask such questions, starting with "Where . . ." or "In which city . . ." etc. Likewise, there are several ways in which the answer can be embedded in the context of a passage, where the answer

- Almost always has token/s starting with an uppercase letter and continues with lowercase letters
- Is almost always flagged as a noun or noun phrase by a POS tagger
- Tends to be preceded by *in* or *at*
- Is often a specialization ("hyponym") of the compound noun *geographical location* in WordNet.

Children are known to exploit such clues during vocabulary

[*] Work done while visiting CMU.

[1] `http://wordnet.princeton.edu/`

acquisition [2]. Similar comments hold for most common types of factual questions. Note that accumulation of "soft" evidence for a match can happen at both the question and the answer end.

To be sure, machine learning is already heavily used inside many QA modules, such as part-of-speech (POS) and named entity (NE) taggers, and shallow parsers. These modules demand training data for different tasks in diverse forms, so it is no wonder that most modules are used as black boxes, pretrained with generic data which we can only hope suffices for our application. We have often seen BBN's Identifinder[2] flag Japanese person names as *organization*; *m.p.h.* and *Gaffney, SC* as *person*; and *telephone*, *all in all*, and *look* as *work of art*. The gazetteer in GATE[3] regards *2000* as possibly representing a year, but not *1803*. Moreover, as described above, these modules are put together via *engineered bias* rather than learning the weight of their evidence for specific QA instances. Clearly, the black-box mode of usage can be improved upon.

Our contributions: To a first approximation, we represent questions and answers using feature vectors in the style of modern information extraction systems like MALLET[4] or Minor Third[5], and then learn high-density regions in the joint distribution of these features. More specifically, given a new query and a candidate answer, we estimate if their combined feature vector is sufficiently close to a high-density region to accept the candidate answer. We argue that many QA ranking strategies are "hard" approximations to such "soft" scoring. As we shall see, the high-density zones can also be conditioned on questions to yield eminently meaningful passage features, which can be used as predictive annotations [17].

While questions are not explicitly classified, entities in passages need to be generalized using an is-a hierarchy. In this work we use WordNet. An interesting feature of our approach is that we do not need to customize WordNet in any way, or solve a WSD problem to attach WordNet to our corpus.

In this paper, we start by showing that useful high-density regions are indeed recoverable from the joint distribution of question and answer features. We then propose a conditional exponential model to predict if there is enough evidence that a candidate passage is likely to answer a given question. We fit the model to features extracted from TREC QA data in conjunction with WordNet, with no customization whatsoever. The model reveals feature correlations which are intuitive and useful in retrospect.

Our trained model can be used to filter candidate response tokens from a passage-level IR system. Experiments with TREC 2000 and TREC 2002 show significant improvements in the earliest rank at which an answer passage is found. Owing to the simplicity of our model, applying our classifier to a passage span is very fast, allowing us to scale up the filtering step to 300 responses from the IR system in 1–2 seconds. Our results move us closer to shrink-wrapped, trainable QA systems which are as easy to deploy as a basic IR system.

2. PRELIMINARIES AND RELATED WORK

In this section we set up our problem more formally, and review some related work.

A factual question is a sequence of **tokens**, such as *Who wrote "20,000 Leagues Under the Sea"*, possibly containing quoted strings (which are regarded as single tokens). A **corpus** is a set of **documents**; each document is a sequence of **sentences**; and each sentence is a sequence of tokens. The QA system returns a ranked list of **passages**. Working definitions of a passage range from a 50- or 250-byte text window, some number of consecutive tokens, or a small number (1–3) of consecutive sentences.

In this work we address queries which seek as answer an **entity** of a **type** that is specified only indirectly, where the entity satisfies some **constraints** expressed in the question. We call the desired answer type the **atype** of the question. In the question above, the answer entity is Jules Verne, the atype is *person*, and the constraint is that the answer entity must have written the specified book.

The constraint is expressed in two parts. First, there are ground constants (like the book title) which we expect to occur essentially unchanged in an answer passage. Constants are matched reasonably easily using an IR system. But enforcing the second part, the *relation* "wrote," generally requires a parse and quite some engineering; this is where many QA systems avoid "NLP completeness" for simplicity and speed. At this stage, we will do likewise, and think of the IR probe as being derived from the template

```
FIND x NEAR GroundConstants(question)
    WHERE x IS-A Atype(question)
```

For starters, `NEAR` may mean linear proximity in text, and as we see later, this is already surprisingly effective. But the stage is wide open for introducing diverse evidence of nearness, including chunking and parsing, and this is an exciting area of future work. The focus of this paper is the predicate

```
x IS-A Atype(question).
```

Most QA systems use a two-step approach to evaluate this predicate. First, they build a *question classifier* to evaluate `Atype(question)`. This classifier is usually completely supervised, and uses a taxonomy of question types [20, 24]. Second, they map the estimated atype to tokens expected to be found in an answer passage [17, 19]. Notable departures from the standard paradigm include inferring paraphrases [13], predicting answer tokens from question features [1, 6], and using a language model conditioned on the question type [23]. A key feature of our approach is that we do not have any fixed system of atypes, and we do not know how to materialize `Atype(question)`. Instead, we evaluate the soft predicate `DoesAtypeMatch(x,question)`.

3. FEATURES

Each token in a question or passage may be associated with a variety of *features*. We will use *surface* features such as orthography and specific word forms, and *taxonomy* features which help us recognize that a passage token is an instance of the atype that the question seeks. Inspired by Yarowsky and others, we will adopt the "kitchen sink" approach [22]: include as many useful-looking features as we can find, use learning techniques robust to redundant features, and exploit the ones that float to the top.

Our approach is to generalize each passage token to many

[2] `http://www.bbn.com/speech/identifinder.html`

[3] `http://gate.ac.uk`

[4] `http://mallet.cs.umass.edu`

[5] `http://sourceforge.net/projects/minorthird`

levels of detail, in a redundant fashion. If a token is a noun found in WordNet, the hypernym paths (definitions are in the next section) to noun roots provides one kind of generalization. Words not found in WordNet are generalized to surface context patterns.

Individually, each feature thus derived can be unreliable. E.g., both place and person names tend to start with an uppercase letter and continue with lowercase letters. But person names are rarely preceded by a preposition. Our hope is that a robust learning algorithm can combine such noisy evidence to give high accuracy.

3.1 Lexical network-based features

We start this section with a brief WordNet primer. For our purposes, WordNet [15] is a graph where nodes are concepts and edges are relations between concepts. A concept is called a **synset** because it is described by a set of synonyms, also called **lemmas**. A synset may be described by many lemmas. Conversely, a lemma (like *match*) can describe many synsets, in which case it is highly **polysemous**. A lemma, a part-of-speech, and a (standardized) sense number together defines a synset uniquely, and is written as `match#n#1` (first noun sense of *match*). We only consider **hypernym** and **hyponym** edges in WordNet, which represent IS-A relations. E.g., in this chain of generalization:

> horse, Equus caballus → equine, equid → $\cdots$ → ungulate, hoofed mammal → placental, placental mammal, eutherian, eutherian mammal → mammal → $\cdots$ → animal, animate being, beast, brute, creature, fauna → $\cdots$ → entity

beast and *brute* are synonyms, *equid* is a **hyponym** of *ungulate*, *horse* is a **hyponym descendant** of *mammal*, *equid* is a **hypernym** of *horse*, and *entity* is a **hypernym ancestor** of *horse*.

Each passage token is mapped to one or more synsets in the WordNet noun hierarchy. Holding extreme faith in the kitchen sink approach (and mainly because it would be computationally expensive), we perform no word sense disambiguation. From each synset we walk up the noun hypernym hierarchy and collect all hypernym ancestor synsets as features associated with the given token. We do not consider non-noun hypernym hierarchies because they are known to not be as usable as the noun hierarchy, and because most answers are nouns or noun-related entities.

WordNet is only a representative example; we can use the occurrence of a token in any precompiled dictionary, and any available sources of instance-of and part-of information as features. E.g., in a medical application, we can use the extensive MeSH hierarchy, and if wanted to support QA on product reviews, we might use the UNSPSC product taxonomy[6].

3.2 Surface context features

Lexical features work only for finite domains, and surface context features are one way to supplement lexical features to handle infinite domains (e.g., quantities).

Surface features for questions currently include (lowercased) token sequences of length up to **three** tokens, starting from standard wh-word question leaders: *when*, *what*, *how*, *where*, *which*, *who*, *whom*, *whose*, *why*, *name*, and *define*. This set is language-specific, and is designed to capture clues about the desired atype.

Surface features of passage tokens are extracted in a manner similar to information extraction systems. We flag if a token has some uppercase letter (`hasCap`), is all uppercase letters, is an abbreviation (uppercase letters and periods), has an uppercase letter followed by two or more lowercase letters (`hasXxx`), has a digit (`hasDigit`), and is entirely digits. (Later, we intend to use the part of speech of the token and its adjacent tokens; our current POS tagger is not as fast as our indexer.)

3.3 Proximity features

Most QA systems reward linear proximity between candidate answers and matched question tokens, and some provide a reasonably formal justification [5]. In our setup, when we consider a passage token as (part of) a candidate answer, we also measure the linear distance (number of intervening tokens) between the candidate token and each matched question token. To make proximity features compatible with all other boolean features, we combine the **reciprocal** of the distances, in two ways: take their **average** or take their **maximum**.

4. LIKELIHOOD RATIO TESTS

We start with some exploratory data analysis: are there indeed consistent, detectable patterns that can be extracted from the feature classes we collect? We limit our study to pairs of features, one from the question and one from the answer. Searching for larger correlated itemsets would be prohibitive in such high dimensional spaces.

Figure 1 shows some pairs of features (only surface features from question and passage tokens) that pass the well-known likelihood ratio test with the highest scores, showing strong association between the feature pairs.

Q-feature	A-feature	Score	
whom	hasCap	159	
whom	hasXxx	162	
who_wrote	hasCap	164	
where_are	hasXxx	166	
who_wrote	hasXxx	167	
who_invented	hasXxx	181	
who_invented	hasCap	200	
what_year	hasDigit	280	
what_city	hasCap	373	
what_city	hasXxx	378	
how_many	hasDigit	449	
of	hasXxx	480	*
what	hasXxx	540	*
what	hasCap	555	*
name	hasXxx	897	
name	hasCap	911	
is	hasXxx	1110	*
is	hasCap	1115	*
where	hasXxx	1196	
where	hasCap	1205	
when	hasDigit	1367	
who	hasXxx	1594	
who	hasCap	1707	

Figure 1: The most strongly associated pairs of surface features in questions and answer tokens are highly intuitive. The score shown is the standard $-2\log\Lambda$ score which is χ^2-distributed. (Meaningless and/or useless pairs are marked "*" and commented upon in the text.)

Can we simply retain some of the top-scoring pairs in Figure 1 in a question-to-answer mapping table and achieve our objective? Perhaps, but any score threshold would be rather arbitrary. A χ^2 table shows almost *all* pairs as significantly associated. Even pairs which are meaningless or too general to be useful, like (who, hasDigit), (what, hasXxx),

[6] `http://www.eccma.org/unspsc`

(what, hasCap), and (is, hasXxx), get significant scores (see "*"). The problems are obvious: the likelihood ratio test does not discriminate among token labels, and does not capture redundancies in correlation information across different pairs.

Q-feature	A-feature	Score
who_painted	artist#n#1	12
how_much	definite_quantity#n#1	24
how_much	metric_unit#n#1	25
how_far	nautical_linear_unit#n#1	26
how_far	mile#n#1	26
how_much	metric_weight_unit#n#1	33
how_many	definite_quantity#n#1	37
how_many	number#n#2	55
how_many	large_integer#n#1	60
when_is	time_period#n#1	79
when_is	calendar_day#n#1	98
who_wrote	writer#n#1	103
when_is	day#n#4	135
city	state_capital#n#1	236
what_city	state_capital#n#1	236
king	sovereign#n#1	275
where_is	region#n#3	283
where	location#n#1	288
king	king#n#1	294
who	person#n#1	316
where	region#n#3	370

Figure 2: Strong associations between surface features of questions and WordNet-derived features of answers are also intuitive. Synsets are written in the standard lemma#pos#sense format.

In other words, a measure of association does not, in itself, tell us how to combine evidence from multiple pairs, because they are often highly redundant. Apart from redundancy among surface features, there is much redundancy among IE and WordNet features. Consider Figure 2, which shows strongly associated pairings between surface features in the question and WordNet synsets derived from the answer passage. These are also by-and-large meaningful, but it is not clear how to discount evidence from one source against evidence from another, based purely on measures of association.

5. PASSAGE SCORING AND FILTERING

Current practice in mapping questions to likely answer passages amounts to the following procedure. Given a question q, extract from it a suitable feature vector $\vec{q}$. Using precompiled mapping tables (and/or an automatic classifier) predict some properties of a feature vector $\vec{p}$ that is likely to be generated by an answer passage, and then look for passages p that generate or satisfy $\vec{p}$.

Here we seek to avoid a fixed classification of $\vec{q}$ by inducing a two-class (yes/no) classifier on the joint distribution of $\vec{q}$ and $\vec{p}$. I.e., we concatenate $\vec{q}$ and $\vec{p}$ into a single feature vector $\vec{x} = (\vec{q}, \vec{p})$, and let y be a boolean prediction. Many classifiers will output not only a boolean prediction but also a score (which may be an estimated probability of belonging to the "yes" class). We can use the score in at least two ways:

Rerank: Use the score given by the classifier to rerank passages, ignoring their original rank as returned by the IR engine.

Eliminate: The original IR ranking is retained, but passages with scores less than a threshold are eliminated, reducing the ranks of true answer passages.

5.1 Non-linear learners

A linear classifier such as a linear SVM will fit a weight vector $\vec{w} = (\vec{w}_q, \vec{w}_p)$ from the training data, and then predict $y \in \{-1, +1\}$ as $\text{sign}(\vec{w} \cdot \vec{x}) = \text{sign}(\vec{w}_q \cdot \vec{q} + \vec{w}_p \cdot \vec{p})$. A linear classifier is unlikely to be suitable. Ideally, we would like different questions to substantially modulate the kinds of features we should seek in passages. But, in case of a linear classifier, fixing the question merely materializes the $\vec{w}_q \cdot \vec{q}$ part into a constant, and the portion $\vec{w}_p$ that scores $\vec{p}$ remains unchanged. Thus the discriminant will assign static notions of positive or negative importance to passage features, irrespective of the question.

The study of pairwise likelihood ratio suggests that our model should directly capture pairwise interactions between features. In kernel classifiers, this can be done using a **quadratic** kernel $K(\vec{x}_1, \vec{x}_2) = (\vec{x}_1 \cdot \vec{x}_2 + 1)^2$, which will potentially capture all pairwise interaction of attributes. For training instances x_i, $1 \le i \le n$, the SVM will estimate dual variables α_i, $1 \le i \le n$, mostly zero, such that the prediction for a new feature vector $\vec{x} = (\vec{q}, \vec{p})$ is $\sum_i \alpha_i y_i K(\vec{x}_i, \vec{x})$.

Because the α_i's, $\vec{x}_i$'s and y_i's are known, given $\vec{q}$ (the question part of $\vec{x}$), we can "materialize" the discriminant $\sum_i \alpha_i y_i K(\vec{x}_i, \vec{x})$ explicitly as a linear (because $\vec{q}$ is known and fixed) function of $\vec{p}$. E.g., if $\vec{q}$ switches on the feature how_many, we expect the coefficient of the features hasDigit and number#n#2 to be large in a positively-labeled $\vec{p}$, and we can use this information to look for and score promising passages more effectively.

5.2 A conditional exponential model

In our prototype, we retain the discriminative learning strategy, but we use a conditional exponential model with our feature-pairs explicitly materialized instead of a SVM with a quadratic kernel. In our experience, the exponential model (a log-linear learner) is faster to train given the very large number of instances created in our QA application. Moreover, in this application, SVM cannot really give us the benefit of kernelization, because during query time, the kernel has to be materialized in any case.

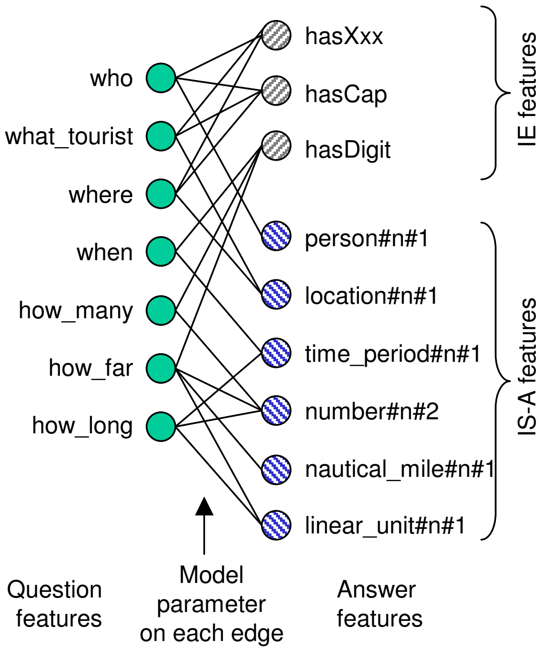

Figure 3: Our model potentially assigns a parameter to each pair of features, one derived from the question and the other derived from the answer.

We explicitly model pairwise feature interactions between feature vectors $\vec{q}$ and $\vec{p}$ by assigning one model parameter for each pair of scalar features, one (q_i) drawn from the question and the other (p_j) drawn from the passage, as shown in Figure 3. We express our model as

$$\begin{aligned} \Pr(Y=+1|\vec{q},\vec{p};\vec{w}) &= \text{logit}\left(w_0 + \textstyle\sum_{i,j} w_{i,j} q_i p_j\right), \\ \text{where} \quad \text{logit}(a) &= e^a/(1+e^a). \end{aligned} \qquad (1)$$

and $\Pr(Y=-1|\vec{q},\vec{p};\vec{w}) = 1 - \Pr(Y=+1|\vec{q},\vec{p};\vec{w})$.

Our overall goal is to maximize the total conditional (log) probability over all $(\vec{q},\vec{p})$ instances over choices of parameters $\vec{w}$, i.e.,

$$\vec{w}^* = \arg\max_{\vec{w}} \sum_{(\vec{q},\vec{p});y} \log \Pr(y|\vec{q},\vec{p};\vec{w}). \qquad (2)$$

To avoid over-fitting and improve numerical stability, we use a Gaussian prior in our optimization [4]:

$$\max_{\vec{w}} \left\{ \sum_{\vec{q},\vec{p},y} \log \Pr(y|\vec{q},\vec{p};\vec{w}) - \gamma \sum_{i,j} w_{i,j}^2 \right\}$$

involving a smoothing parameter γ which is chosen by cross-validation. Experimental data suggests that careful tuning of γ can help the log-linear model compete favorably with SVMs [25]. We use L-BFGS, a sparse, limited-memory, quasi-Newton numerical solver[7] to search for the best w.

Note that each parameter $w_{i,j}$ (except the offset w_0 and parameters associated with proximity features, which we avoid discussing for simplicity) is associated with one question feature and one answer feature. Therefore, inspection of these parameters can reveal interesting and possibly unexpected structures in the relationship between question and passage features.

```
Index sliding passage windows of three sentences using an IR system.
Randomly partition QA pairs for training and testing (5–10 folds were used).
for each training question $q$ do
    Collect question features into a feature vector $\vec{q}$ as described.
    Send the question to the IR system and get top 300 response passages
    Using the regular expressions provided by TREC, mark token spans where the correct answer appears. Keep each answer span as tight as possible.
end for
for each token over all passages do
    Collect surface features from token.
    Map token to noun synset(s) in WordNet if possible, and collect all hypernym ancestors.
    Together, these features give $\vec{p}$, the feature vector derived from the passage.
    $\vec{q}$ and $\vec{p}$ together define an instance.
    The label $y$ for the instance is $+1$ if the passage token is inside an answer span, and $-1$ otherwise.
end for
```

Figure 4: Collection of training instances for the log-linear classifier.

6. EXPERIMENTS

We experimented with two years of TREC QA data (2000 and 2002) and got broadly similar results. The corpus was chopped up into sentences. Three consecutive sentences were defined as a passage. Passages were indexed using Lucene[8], which implements a standard TFIDF search.

```
Collect question features into a feature vector $x_q$ as described.
Send the question to the IR system and get top 300 response passages
for each token over all candidate passages do
    Obtain passage features $x_p$ and combined feature vector $x$ as during training
    Using the log-linear classifier, get a boolean prediction for the token
end for
if all tokens in a passage are eliminated then
    Eliminate the passage
end if
Present surviving passages in the order originally provided by the IR system.
```

Figure 5: Using the classifier to prune the passage list.

The training process is shown in Figure 4, and the testing process is shown in Figure 5. TREC 2000 yielded a training set with 198496 instances and 674631 (largely boolean) features, where the average instance had only 42 features turned on. For TREC 2002 the corresponding numbers were 170264 instances, 540901 features and average fill of 43.6. Convergence was declared when $\|\nabla(w)\|/\|w\|$ dropped below 0.001; this happened between 500 and 2000 iterations (10–40 minutes on a 1GHz 1GB P4) depending on γ.

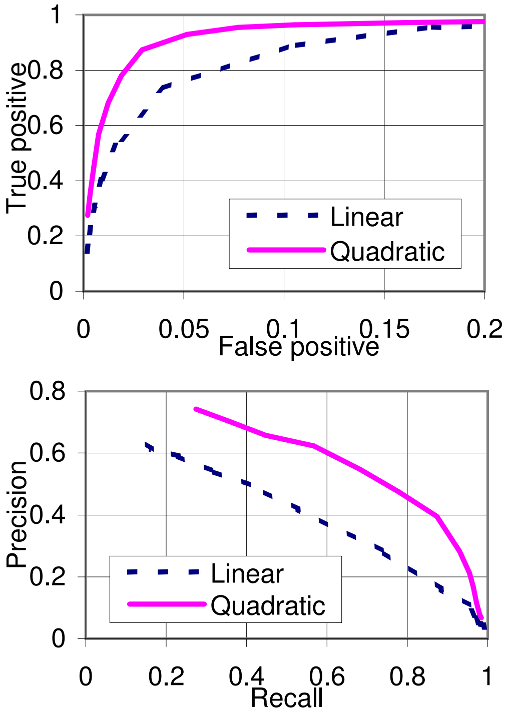

Figure 6: The quadratic representation enables a much better fit to the data and consequent accuracy on test data. ROC and recall-precision curves are shown. The breakeven F1 value is 0.44 for the linear model and 0.61 for the quadratic model.

There are two related design choices during testing where more exploration is warranted in future work. First, we

[7] `http://riso.sourceforge.net`

[8] `http://jakarta.apache.org/lucene/`

eliminate passages rather than *rerank* them (see §5). We also tried reranking, but the actual value of the logistic score was not reliable enough. Second, a passage survives if any of its tokens cannot be eliminated. I.e., token scores are not combined into a passage score in any sophisticated fashion.

6.1 Linear vs. quadratic

For many applications, dependencies between features are not well-understood, and the choice of a kernel is an experimental art. In our application, however, verifying that a quadratic representation gave much greater accuracy was an important part of vindicating our model. Figure 6 confirms our intuition. In both ROC and recall-precision measures, the quadratic model is far superior.

Our positive experience with the quadratic representation may have implications for setting up log-linear models in IE and shallow parsing. Efforts to synthesize non-linear combinations of raw features to strengthen the log-linear model in these domains are relatively recent [14].

A. where		**B. who**	
hasXxx	0.27	hasXxx	0.38
region#n#3	0.25	hasCap	0.31
location#n#1	0.25	entertainer#n#1	0.16
hasCap	0.18	leader#n#1	0.14
country#n#1	0.17	artist#n#1	0.14
district#n#1	0.17	performer#n#1	0.13
hasDigit	-0.1	person#n#1	0.09
artifact#n#1	-0.13	poet#n#1	0.08
measure#n#3	-0.14	location#n#1	-0.18
abstraction#n#6	-0.16	entity#n#1	-0.21
act#n#2	-0.16	hasDigit	-0.21
		abstraction#n#6	-0.31

C. when		**D. how_many**	
hasDigit	0.65	hasDigit	0.16
time_period#n#1	0.08	integer#n#1	0.07
holiday#noun#2	0.06	*twelve#n#1	0.06
day#noun#4	0.06	number#n#2	0.06
calendar_month#n#1	0.05	definite_quantity#n#1	0.06
calendar_day#n#1	0.05	measure#n#3	0.03
hasXxx	-0.09	cognition#n#1	-0.03
hasCap	-0.12	relation#n#1	-0.03
object#n#1	-0.13	object#n#1	-0.05
entity#n#1	-0.18	act#n#2	-0.05
		entity#n#1	-0.07

E. how_far		**F. linear_unit#n#1**	
nautical_mile#n#2	0.02	how_far	0.02
hasDigit	0.02	*where_is	0.009
linear_measure#n#1	0.02	*what_speed	0.005
linear_unit#n#1	0.02	how_long	0.003
measure#n#3	0.01	whom	-0.002
hasXxx	-0.004	when	-0.004
hasCap	-0.005	what_city	-0.004
object#noun#1	-0.006	how_many	-0.005
entity#n#1	-0.007	what	-0.007

G. location#n#1		**H. hasDigit**	
where	0.249	when	0.65
city	0.109	what_year	.18
province	0.029	how_many	0.16
country	0.015	how_much	0.09
state	0.012	which_date	0.05
*tourist	0.004	*how_hot	0.02
year	-0.0230	how_far	0.02
how	-0.0314	company	-0.03
when	-0.043	city	-0.05
name	-0.113	name	-0.09
who	-0.178	where	-0.10
		who	-0.21

Figure 7: Fixing features of questions and watching answer parameters, and vice versa. Items marked with a "*" are meaningful, but were unanticipated from our limited study of the corpus and question-answer pairs.

6.2 Model parameter anecdotes

Each component (except for w_0) of the parameter vector w estimated by the classifier corresponds to a question feature and a passage feature. We can set a feature in the question (respectively, answer), sort the model parameters, and look for the answer (respectively, question) features with the smallest and largest coefficients.

Figure 7(A–E) shows the results. *Where* and *who* questions share hasCap and hasXxx, but differentiate answers based on WordNet features. Similarly, the sharing of hasDigit does not confound *when*, *how_many*, and *how_far*.

In all cases, very generic features (e.g., the noun roots) get negative coefficients. Why not zero? A little thought shows this is necessary for rejecting false positives. As a by-product, the sign may help us decide which predictive annotations to index.

In (F–H) we set *answer* features to reveal abstraction of question features. Again, we see intuitive question classes which require answers of the set type.

6.3 Rank improvement via filtering

As shown in Figure 5, passages are considered in the order returned by Lucene. We scan the top response passages, convert each token to a feature vector, and subject the token to the trained classifier. The classifier returns a score $\Pr(Y = +1|x)$ for each token. We eliminate tokens that have scores lower than a fixed threshold and eliminate a passage if all its tokens are eliminated. Surviving passages are presented in their original order. If the passage contains the TREC-specified answer as a substring, it is an answer passage.

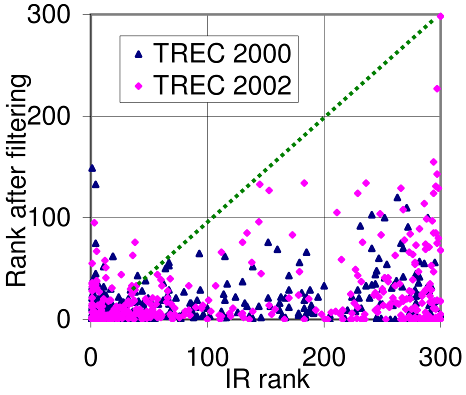

Figure 8: Scatter plot of ranks of first correct passages before (x) and after (y) filtering. Points below the diagonal are good news.

In Figure 8 we plot a scatter of ranks before and after filtering. We see that filtering is very effective at improving the ranks of answer passages, often dropping them from 200–300 down to less than 50. The comparison with the IR baseline demonstrates that our model is good at discriminating answer tokens from non-answer tokens.

Why use an IR baseline which is worse than the best QA systems reported? Ideally, we should start with the best system, enhance it with our new feature representation and learning ideas, and make standard end-to-end MRR5 measurements. In practice, hardly any QA system is available for downloading, and even demos are rare. LCC[9], among

[9] http://www.languagecomputer.com/

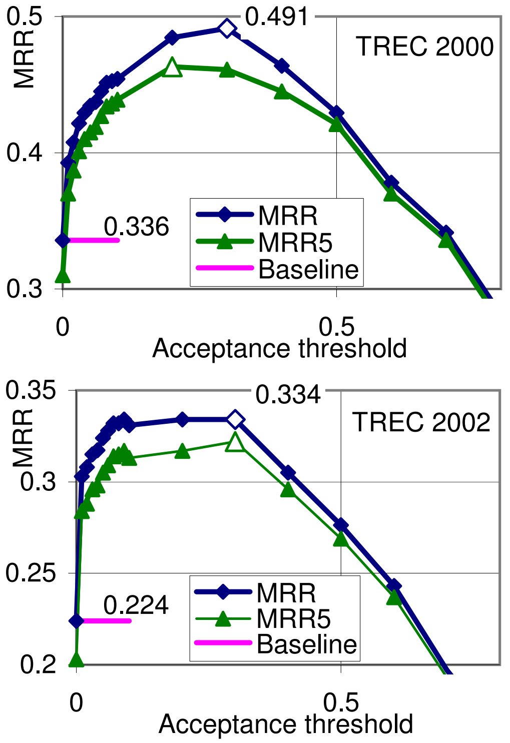

Figure 9: By using a suitable threshold on $\Pr(Y = +1|x)$, we can prune away large number of non-answer passages and increase MRR beyond the IR baseline (threshold=0).

the best QA systems, is an exception, but the exact corpus is not known and the results must be scraped from HTML. Unless code is available, it is impossible to replicate the myriad undocumented details of tokenization, compound word detection, chunking, POS tagging, query formulation, and so on.

For all these reasons, it is very common [10, 21] for researchers to use an IR baseline, change very few things at a time, and make careful assessments of the effect of each enhancement. Katz and Lin [10] claim that an extensive recent study [12] presents "substantial empirical evidence that boolean-based passage retrieval techniques are sufficient to obtain reasonable performance in question answering tasks."

In the QA literature the **mean reciprocal rank (MRR)** is commonly used as an aggregated figure of merit. Suppose $n_q \geq 1$ is the earliest rank of the passage at which the answer to question $q \in Q$ is found. Then MRR is defined as $(1/|Q|)\sum_q(1/n_q)$. MRR is between 0 and 1, and a higher MRR is better. TREC contestants are required to report MRR5, where any value of $n_q > 5$ must be regarded as ∞.

MRR and MRR5 (while perhaps reasonable as final figures of merit) offer little guidance in diagnosing the strengths and weaknesses of the many components of QA systems. MRR assigns the same penalty for dropping an answer from rank 1 to rank 2 as the penalty for dropping the answer from rank 2 to rank ∞ (i.e., not reporting it at all). MRR5 will ignore a rank reduction from 20 to 6. In relative terms, MRR5 will be more hostile to a rank reduction from 20 to 6 than a rank reduction from 20 to 10. Both MRR numbers conflate many critical measurements into single end-to-end numbers.

In Figure 9 we plot MRR and MRR5 against the posterior probability threshold at which a passage token is accepted as (part of) an answer. As we assert larger acceptance thresholds, we filter out non-answers and increase MRR beyond the IR baseline. Beyond some acceptance threshold, we get too demanding and start losing the true answer tokens. As can be seen, the peak MRR is substantially larger than the IR baseline (+46% for TREC 2000, +49% for TREC 2002). MRR and MRR5 differ only slightly; in many cases we do not get an answer because there *is* no answer in the top 300 passages, and query expansion or back-off techniques will likely help. To give an impression of where we stand relative to highly-engineered champion systems, for TREC 2000, where our MRR5 is 0.463, the top scores are 0.76 (LCC again), 0.46, 0.46, and 0.31. Our "clean-room" approach to QA is quite promising, at least considering rank order. (We could not obtain MRR5 scores for TREC 2002.)

6.4 Ablation and drill-down studies

We also did ablation experiments with features. For TREC 2002, filtering passages with the *known* BBN Identifinder tag gave us an MRR which was several percent lower. Clearly, the exponential model is learning something extra from all those features. As Figure 10 shows, features derived from WordNet turn out to be more important than surface patterns, but for one of our two data sets, surface patterns have visible marginal benefits.

Features	Recall	Precision	F1
	TREC 2000		
Surface	0.024	0.497	0.047
WordNet	0.427	**0.634**	0.510
Surface+WordNet	**0.447**	0.619	0.519
	TREC 2002		
Surface	0.035	0.492	0.066
WordNet	0.383	**0.662**	0.485
Surface+WordNet	**0.451**	0.653	**0.533**

Figure 10: Effect of including/excluding feature subsets.

Figure 11 shows how MRR gains for specific question types compared with the average gain (scaled to 1). The most visible deviations from the mean are for *what*, *which* and *who*. Questions with *who* are doing very well by combining evidence from orthography and WordNet. Questions with *what* and *which* are doing relatively poorly (even as the absolute improvement is 30–40%).

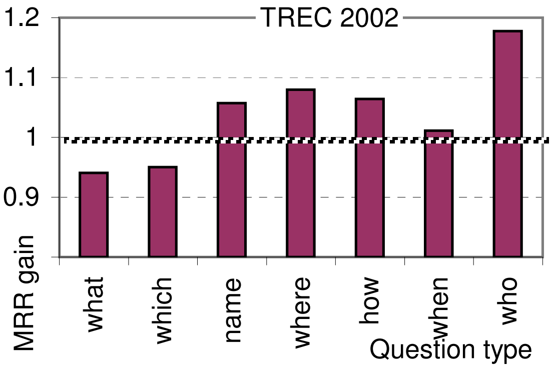

Figure 11: MRR gain ratio of various question types relative to average overall MRR gain ratio.

We found two distinct factors that explain our observation. First, sequences of 1–3 tokens starting with *what* are not enough to capture enough clues about the atype. Second, for both *what* and *which* questions, the atype space is, in principle, unbounded. In ongoing work, we are adding a synthesized "test for equality" feature for *what* and *which* questions. Summarizing, we feel that we can retain the simplicity of our basic approach and perform better with some more work on feature extraction and representation.

7. CONCLUSION

We have presented a very simple but promising exponential model coupling question and passage features for the QA problem. Our results move us closer to shrink-wrapped, end-user trainable QA systems. We believe that such indirect supervision of language tasks will become increasingly important, and components must plug into a global probabilistic infrastructure rather than be engineered black boxes.

Our work opens up some obvious avenues for exploration. We should explore higher order kernels with maximum margin learners. We should improve feature extraction, especially for *what* and *which*, possibly using chunking on the question. It would be nice to integrate disambiguation naturally into the framework, rather than as a preprocessing step. Finally, it would be promising (but computationally challenging) to extend the flat feature space in our exponential model to a more elaborate graphical model involving some salient parts of the lexical network.

8. REFERENCES

[1] E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. In *WWW Conference*, pages 169–178, 2001.

[2] P. Bloom. *How Children Learn the Meanings of Words*. MIT Press, Cambridge, MA, 2000.

[3] E. Breck, J. Burger, D. House, M. Light, and I. Mani, editors. *Question Answering from Large Document Collections*, AAAI Fall Symposium on Question Answering Systems, 1999.

[4] S. F. Chen and R. Rosenfeld. A gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Carnegie Mellon University, 1999.

[5] C. L. A. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *SIGIR*, pages 358–365, 2001.

[6] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In *SIGIR*, pages 291–298, 2002.

[7] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in KnowItAll. In *WWW Conference*, New York, 2004. ACM.

[8] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. FALCON: Boosting knowledge for answer engines. In *TREC 9*, pages 479–488, Gaithersburg, MD, 2000. NIST.

[9] E. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C.-Y. Lin. Question answering in webclopedia. In *TREC 9*, Gaithersburg, MD, 2000. NIST.

[10] B. Katz and J. Lin. Selectively using relations to improve precision in question answering. In *EACL Workshop on Natural Language Processing for Question Answering*, Budapest, Hungary, 2003.

[11] C. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the Web. In *WWW*, volume 10, pages 150–161, Hong Kong, may 2001. IW3C2 and ACM. See `http://www10.org/cdrom/papers/120/`.

[12] M. Light, G. Mann, E. Riloff, and E. Breck. Analyses for elucidating current question answering technology. *Journal of Natural Language Engineering*, 7(4):325–342, 2001.

[13] D. Lin and P. Pantel. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[14] A. McCallum. Efficiently inducing features of conditional random fields. In *UAI*, 2003.

[15] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Introduction to WordNet: An online lexical database. *International Journal of Lexicography*, 1993.

[16] E. Nyberg, T. Mitamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-Thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, S. Judy, J. Ko, A. Kupsc, L. V. Lita, V. Pedro, D. Svoboda, and B. V. Durme. The JAVELIN question-answering system at TREC 2003: A multi-strategy approach with dynamic planning. In *TREC*, volume 12, 2003.

[17] J. Prager, E. Brown, A. Coden, and D. Radev. Question-answering by predictive annotation. In *SIGIR*, pages 184–191. ACM, 2000.

[18] D. Radev, W. Fan, H. Qi, H. Wu, and A. Grewal. Probabilistic question answering on the web. In *WWW Conference*, pages 408–419, 2002.

[19] G. Ramakrishnan, S. Chakrabarti, D. A. Paranjpe, and P. Bhattacharyya. Is question answering an acquired skill? In *WWW Conference*, pages 111–120, New York, 2004.

[20] J. Suzuki, T. Hirao, Y. Sasaki, and E. Maeda. Hierarchical directed acyclic graph kernel: Methods for structured natural language data. In *ACL*, pages 32–39, 2003.

[21] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 41–47. ACM Press, 2003.

[22] D. Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *ACL*, volume 32, pages 88–95, Las Cruces, NM, 1994.

[23] D. Zhang and W. S. Lee. A language modeling approach to passage question answering. In *Text REtrieval Conference (TREC)*, volume 12, Gaithersburg, MD, Nov. 2003. NIST.

[24] D. Zhang and W. S. Lee. Question classification using support vector machines. In *SIGIR*, Toronto, Canada, 2003. ACM.

[25] J. Zhang and Y. Yang. Robustness of regularized linear classification methods in text categorization. In *SIGIR*, pages 190–197. ACM, 2003.

[26] Z. Zheng. AnswerBus question answering system. In *Human Language Technology Conference (HLT)*, 2002.