The Use of Predictive Annotation for Question Answering in TREC8

John Prager, Dragomir Radev, Eric Brown, Anni Coden
IBM TJ Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
{jprager,radev,ewb,anni}@us.ibm.com

Valerie Samn
Teachers College
Columbia University
New York, NY 10027
vs115@columbia.edu

ABSTRACT

This paper introduces the technique of Predictive Annotation, a methodology for indexing texts for retrieval aimed at answering fact-seeking questions. The essence of the approach can be stated simply: index the answers. This is done by establishing about 20 classes of objects that can be identified in text by shallow parsing, and by annotating and indexing the text with these labels, which we call QA-Tokens. Given a question, its class is identified and the question is modified accordingly to include the appropriate token(s). The search engine is modified to rank and return short passages of text rather than documents. The QA-Tokens are used in later stages of analysis to extract the supposed answers from these returned passages. Finally, all potential answers are ranked using a novel formula, which determines which ones among them are most likely to be correct.

1. INTRODUCTION

For question-answering, system designers have the choice of using technology from Information Retrieval or Natural Language Processing, or some combination thereof [1,3,4].

Information Retrieval systems employing traditional search engines are efficient but suffer from the fact that they generally return documents rather than answer passages, let alone precise answers, and that the documents that are returned are ranked based on frequency of occurrence of query terms rather than any correspondence with what the query is seeking. Natural Language Processing systems can to a greater or lesser extent overcome the problem of semantic matching, but are inherently expensive; this can make processing a database the size of that in the TREC8 exercise inherently intractable.

Our approach attempts a middle ground. Our group's principal experience has been with traditional IR systems, but also with building text-analysis systems such as TEXTRACT [6,7].

We decided to build a modified search engine that works in conjunction with shallow NLP of the text. We call our technology Predictive Annotation since we identify and annotate in the text generalizations of the base terms; these annotations are designed to correspond to the terminology used in questions.

Our approach is based on the following five observations of questions seeking facts (as opposed to How and Why questions that seek procedural answers) and the texts that typically contain them.

(1) In documents that contain the answers, the query terms that occur there tend to occur in close proximity to each other. They will typically occur within passages of 2-3 sentences - often within one sentence. It is only the single occurrence of a query term in this passage that seems to count; other occurrences elsewhere are more-or-less irrelevant.

(2) The answers to fact-seeking questions are usually phrases: noun phrases ("President Clinton"), prepositional phrases ("in the mountains") and adverbial phrases ("today").

(3) These phrases can be typed by a set of a dozen or so labels (such as PERSON$, PLACE$, MONEY$, LENGTH$,...).

(4) These categories correspond to question words ("Who", "Where", "How much", "How long", ...).

(5) The phrases can be identified in text by simple pattern-matching techniques.

In this paper, we describe the different stages that our system goes through to select answers and we present our results on the official TREC evaluation [1].

2. SYSTEM OPERATION OVERVIEW

It will be clear by now that phrases play a prominent role in our system. It is no coincidence that, for the most part, phrases can be detected in text with a relatively simple pattern-matching algorithm, certainly a requirement that falls far short of full natural-language understanding.

The implementation of the solution is expressed in modifications to three of the components of a traditional search engine solution, namely query-analysis, text parsing and indexing, and scoring.

(1) The user queries are pre-processed with question-words replaced by an invented set of labels we call QA-Tokens.

(2) The text to be indexed is analysed for phrases of certain types. The QA-Tokens corresponding to these types are indexed in addition to the base terms.

(3) The matching process scores short sequences of sentences rather than documents, with weighting much coarser than the traditional tf*idf or its variants.

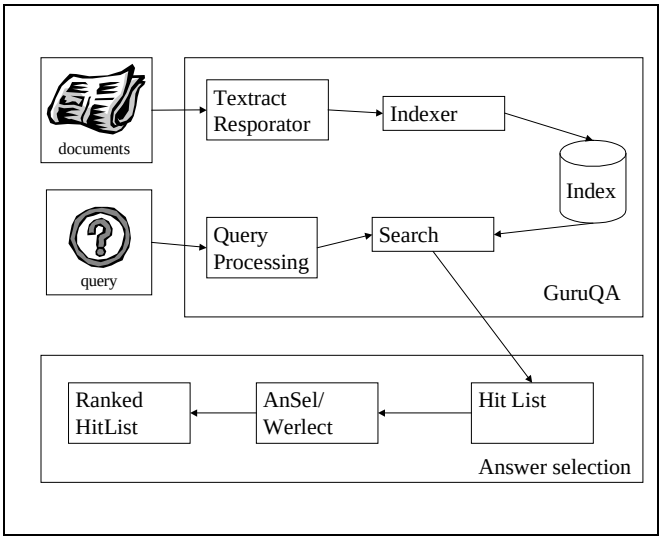

Figure 1: System Architecture

3. QUERY PROCESSING

The query analysis is enhanced by developing a set of question-templates that are matched against the user's query, with substitution of certain query terms with our special QA-Tokens that correspond to the phrase labels mentioned above. So for example, the pattern "where..." causes the word "where" to be replaced with PLACE$. The pattern "how much does ... cost" causes those terms to be replaced with MONEY$. The pattern "how old ..." causes a replacement with AGE$. The base set of such labels is: PLACE$, PERSON$, ROLE$, NAME$, ORGANIZATION$, DURATION$, AGE$, DATE$, TIME$, VOLUME$, AREA$, LENGTH$, WEIGHT$, NUMBER$, METHOD$, MOST$, RATE$ and MONEY$. More specific versions of these, such as STATE$, COUNTRY$, CITY$, YEAR$ can be used as long as the phrase analyser (discussed below) can recognize such quantities.

In some patterns, the entire set of matching terms is removed, as when "How much does <any text> cost" gets transformed to "MONEY$ <any text>". In some other patterns, though, one or more of the matching terms is retained, as when "What is the population of <any text>" gets transformed to "NUMBER$ population <any text>". The set of these patterns currently numbers around 180. The QA-Token set currently numbers around 20. The QA-Tokens are listed in Figure 2, along with the question-words they can correspond to (this mapping is close to, but not exactly, one-to-one) and sample patterns in text they are aiming to discover.

QA-Token	Question type	Example
PLACE$	Where	In the Rocky Mountains
COUNTRY$	Where/What country	United Kingdom
STATE$	Where/What state	Massachusetts
PERSON$	Who	Albert Einstein
ROLE$	Who	Doctor
NAME$	Who/What/Which	The Shakespeare Festival
ORG$	Who/What	The US Post Office
DURATION$	How long	For 5 centuries
AGE$	How old	30 years old
YEAR$	When/What year	1999
TIME$	When	In the afternoon
DATE$	When/What date	July 4th, 1776
VOLUME$	How big	3 gallons
AREA$	How big	4 square inches
LENGTH$	How	3 miles

	big/long/high	
WEIGHT$	How big/heavy	25 tons
NUMBER$	How many	1,234.5
METHOD$	How	By rubbing
RATE$	How much	50 per cent
MONEY$	How much	4 million dollars

Figure 2: QA-tokens

A synonym operator @SYN() is used to deal with cases where a question could be validly matched against more than one type of phrase. Thus a "who" question could match a proper name, a profession or an organization, so will generate @SYN(PERSON$, ROLE$, ORG$, NAME$) in the modified query.

The document collection is analyzed by TEXTRACT () prior to indexing; one of the outputs of this process is a dictionary containing the collection vocabulary. This is used in query processing to discover proper names in the query and optionally to convert names and terms to their canonical forms. A subsystem of TEXTRACT is used to convert common words to their lemma forms, and identify stop-words for removal (which happens AFTER the pattern-matching described above).

We do not weight terms in the index in the traditional IR fashion; instead we weight selected query terms, and specify this in the query syntax by means of a weight operator @WEIGHT(). We use a very coarse granularity of weighting. We choose a base weight of 100 for common words.

Proper names and other multi-word terms in the query are rarer than individual words and so their presence in answer sentences gives more confidence that the sentence is correct than the presence of single terms from the query, all other things being equal, so should be weighted higher. We use a weight of 200 for such items.

Now, any proposed answer text is no answer if it doesn't contain a type-compatible match to a special query-token in the query, so special query tokens should be weighted higher than any other words in the query. We use a weight of 400 for the QA-Tokens.

Some of the alternatives in the @SYN-sets may be more desirable than others. For example, "when" might generate @SYN(DATE$, TIME$), where DATE$ matches specific dates (e.g. "July 4th, 1776") but TIME$ matches more general expressions ("in the afternoon"); a DATE$ match is therefore usually more desirable than a TIME$ match so might be weighted more. We currently don't differentially weight within a @SYN-set. However, we do order the elements in a @SYN-set in decreasing order of desirability; this order is considered in the final answer-selection phase when the passages returned from our search engine contain multiple contenders for "the answer".

We take into account the density of the matching words in a scored passage. Intuitively, since the query words all occur together in a short sentence (the query), so the closer together they occur in a text passage, all other things being equal, the more likely the text reflects the semantics of the query. Hence we calculate a density component to the passage's score in the range 0 to 99 (the latter representing the case of all matching terms being adjacent). This is added to weighted score of appearing query terms.

Finally, the query can be augmented by the @WIN operator through which we specify the target window size and whether matching in this window is to be *exclusive* or not. The window size is an integer representing the size in sentences of a moving window of text within which matching is attempted. We extended this approach to that of using a dynamic window, which uses the stated window size as an upper bound, but prefers sub-windows if they happen to contain the same matches as the larger window. The issue of exclusivity represents the desire to avoid having a QA-Token match a word in the text that is already matched with another query term. Thus if the query is "Whom did President Clinton meet" and the text states that "President Clinton met Tony Blair ...", we don't want the PERSON$ token in the query to match with the PERSON$ attached to President Clinton but rather the one attached to Tony Blair.

4. INDEXING

Our extensions to indexing are somewhat similar to the Predictive Indexing of [4], which adds to the index related terms discovered via WordNet[9]. Both techniques are aimed at increasing recall. Their approach does this by adding related terms to the index, 'in case' the user phrases questions with that particular. Instead, we annotate with the QA-Token that stands for the conceptual category of the index term, and is generated by our query-analysis process.

The indexer runs its own pattern template matcher against the text in the documents to be indexed. For each of the phrase types, a set of patterns needs to be developed.

For example, the following are some of the TIME$ phrases:

in the afternoon

in the morning

...

in :CARDINAL hours

(where :CARDINAL is a cardinal number), and so on. Clearly to avoid a huge list (consider that instead of "hours" in the last example, almost any word indicating a period of time could be substituted), a mechanism for concisely expressing such variants and for efficient performance of the matching is desirable. We built such a system as another annotator for TEXTRACT.

Whenever the indexer succeeds in matching a phrase pattern template in text, the corresponding QA-Token (such as TIME$ or PLACE$) is generated and indexed at that point in the document, along with the individual terms that comprised the phrase. We call this process of adding extra indexing terms annotation. All terms in the document not matched in this way are indexed in the usual way too.

5. SEARCHING

The search engine operates essentially by the usual bag-of-words matching technique, but is subtly affected by the presence of the QA-Tokens. Thus the query:

"When did the Challenger explode" gets translated on query analysis to the bag {@SYN(DATE$, TIME$) Challenger explode} which matches best against locations in the index that contain (exactly or variants of) the word Challenger, the word explode and either a DATE$ or a TIME$ token, meaning some phrasal expression of a time or date.

The QA-Tokens are matched after the other query terms in order to be able to enforce exclusivity. It is not required, though, that there be any QA-Token in the query at all; this is the situation which occurs when the query doesn't match any of our Query Patterns. The search engine operates in such cases just as it would if there were QA-Tokens, except that exclusivity is not an issue.

The final mentioned improvement is to the scoring algorithm. Search engines usually score documents based on how many of the query terms they contain and how often, combining contribution weights computed for each term based on document and collection-based statistics. It is our observation that when a document successfully answers a question, all of the components of the question are to be found together, usually within a passage of a sentence or two. The number of other occurrences of query terms elsewhere in the document does not seem to be a useful indicator of the passage's worth.

Thus we modify the scoring algorithm to score sentences (or short sequences of them) rather than documents. Due to the more severe filtering constraints imposed by this (i.e. that all, or most, query terms must occur in such a passage, rather than the document as a whole), then a less complicated scoring function, namely the weighting scheme described earlier, has been found to suffice.

6. ANSWER SELECTION

The earlier sections of this paper described how we retrieve relevant passages that may contain the correct answer to a query. The output of the search engine at this stage consists of a large number (often more than 30 or 40) potential answers. The following three sections describe how we determine which ones among these answers are more likely.

6.1 Answer ranking

The TREC8 QA-Track requires participants to return sequences of text, which we call here spans, of length either 50 or 250 bytes. This part of the paper describes two systems, AnSel and Werlect, which are used independently of each other to extract these spans from the passages returned by GuruQA, and rank them. AnSel and Werlect use different approaches, which we describe, evaluate and compare and contrast. The output of either system consists of five[1] text extracts per question that contain the likeliest answers to the questions.

GuruQA is first used to find the passages that are considered to be most relevant to the question and labels them with QA-tokens as shown below.

6.2 Sample Input to AnSel/Werlect

The role of answer selection is to decide which among the spans extracted by GuruQA are most likely to contain the precise answer to the questions. Figure 3 contains an example of the data structure passed from GuruQA to our answer selection modules. The example is taken from the official TREC evaluation questions.

```
<p><NUMBER>1</NUMBER></p>
<p><QUERY>Who is the author of the book, "The Iron Lady: A Biography of Margaret Thatcher"?</QUERY></p>
<p><PROCESSED_QUERY>@excwin(*dynamic* @weight(200 *Iron_Lady) @weight(200 Biography_of_Margaret_Thatcher) @weight(200 Margaret) @weight(100 author) @weight(100 book) @weight(100 iron) @weight(100 lady)
```

[1] This parameter is variable and has been set to five empirically.

```
@weight(100 :) @weight(100 biography)
@weight(100 thatcher) @weight(400 @syn(PERSON$
NAME$)) )</PROCESSED_QUERY></p>

<p><DOC>LA090290-0118</DOC></p>

<p><SCORE>1020.8114</SCORE></p>

<TEXT><p>THE IRON LADY; A <span
class="NAME">Biography of Margaret
Thatcher</span> by <span class="PERSON">Hugo
Young</span> (<span class="ORG">Farrar ,
Straus & Giroux</span>) The central riddle
revealed here is why, as a woman <span
class="PLACEDEF">in a man</span>'s world,
<span class="PERSON">Margaret Thatcher</span>
evinces such an exclusionary attitude toward
women.</p></TEXT>
```

Figure 3: Input sent from GuruQA to AnSel

The input consists of four items:

- a query (e.g., "Who is the author of the book "The Iron Lady: A Biography of Margaret Thatcher"?),
- a list of passages (one is shown above; it is surrounded with <TEXT> and </TEXT>),
- a list of annotated text spans within the passages, annotated with span types (QA-tokens), and
- the list of potential span types (or SYN-set) for the type of question recognized by Resporator (e.g., "PERSON$ NAME$" in the example above).

In Figure 3, we only showed the first passage retrieved by GuruQA. It contains five spans, of which three ("Biography of Margaret Thatcher", "Hugo Young", and "Margaret Thatcher") are of types included in the SYN-set for the question (PERSON NAME). The total output of GuruQA for this question includes five passages and a total of 14 potential spans (5 PERSONs and 9 NAMEs).

6.3 Sample Output of AnSel/Werlect

Our system has two outputs: one internal to the system and one that is submitted for evaluation.

6.3.1 Internal output

The internal output is a ranked list of spans as shown in Table 1. It represents a ranked list of the spans (potential answers) sent by GuruQA.

Score	Span
5.06	Hugo Young
-8.14	Biography of Margaret Thatcher
-13.60	David Williams
-18.00	Williams
-19.38	Sir Ronald Millar
-25.80	PP
-26.06	Santiago
-31.75	Oxford
-32.38	Maggie
-36.78	Seriously Rich
-42.68	FT
-198.34	Margaret Thatcher
-217.80	Thatcher
-234.55	Iron Lady

Table 1: Ranked potential answers to Question 1

While only the external output (see below) was required for the TREC evaluation, our system's internal output can be used in a variety of related applications. For example, we can highlight the actual span that we believe is the answer to the question within the context of the passage in which it appears. We can also perform frequency analyses based on the extracted spans.

6.3.2 External output

The external output is a ranked list of 50-byte and 250-byte extracts. These extracts are selected in a way to cover the highest-ranked spans in the list of potential answers. Examples are given later in the paper.

7. ANALYSIS OF CORPUS AND QUESTION SETS

To train our system, we used the set of 38 training questions provided by NIST. In the rest of this paper, we will refer to these questions as TR38.

The results presented in the evaluation section of this paper are based on the 200 test questions (also provided by NIST) which we were not allowed to look at until the official submission of our results.

In this section we describe the corpora used for training and evaluation as well as the questions contained in the training and evaluation question sets.

7.1 Corpus analysis

The corpus used for both training and evaluation (see Table 2) consisted of approximately 2 GB of news articles from four equally represented sources: the Foreign Broadcast Information Service (FBIS), the Los Angeles

Times (LA), the Financial Times (FT), and the Federal Registry (FR). This corpus has been used as a standard in several past TREC text retrieval conferences.

	Year	Size in MB	No. of Docs
FBIS	1996	493	130,471
LA	1989-90	498	131,896
FT	1991-94	592	210,158
FR	1994	414	55,630

Table 2: Description of the corpus used

7.2 Training set TR38

The training set contained questions for which the answers were provided to us for system training and parameter estimation.

Some sample questions are shown in Figure 4:

Question/Answer (TR38)
Q: Who was Johnny Mathis' high school track coach? A: **Lou Vasquez**
Q: What year was the Magna Carta signed? A: **1215**
Q: What two companies produce bovine somatotropin? A: **Monsanto and Eli Lilly**
Q: When did Nelson Mandela become president of South Africa? A: **10 May 1994**

Figure 4: Sample questions from TR38

7.3 Test set T200

Question/Answer (T200)
Q: Why did David Koresh ask the FBI for a word processor? A: At the weekend Mr Koresh requested a word processor to enable him **to record his revelations**.
Q: Who was chosen to be the first black chairman of the military Joint Chiefs of Staff? A: **Colin Powell**
Q: How tall is the Matterhorn? A: The institute revised the Matterhorn 's height to **14,776 feet 9 inches**
Q: How tall is the replica of the Matterhorn at Disneyland? A: In fact he has climbed the **147-foot** Matterhorn at Disneyland every week end for the last 3 1/2 years

Figure 5: Sample questions from T200

The majority of the questions (see Figure 5) in T200 were not substantially different from these in TR38. The introduction of "why" and "how" questions as well as the wording of questions in the format "Name X" caused us some trouble because at the time we had no matching question templates. Other problems were caused by questions that require specific semantic types of answers, such as "star", "designer", "film", and "export" for which our system didn't contain extraction and labeling patterns[2]. Other problems were because of oversights in our question templates and were easily fixed; for example, we were missing a pattern that matched "How rich …" and generated the MONEY$ token.

Some examples of problematic questions are shown in Figure 6.

Q: Why did David Koresh ask the FBI for a word processor?
Q: Name the designer of the shoe that spawned millions of plastic imitations, known as "jellies".
Q: What is the brightest star visible from Earth?

Q: What are the Valdez Principles?

Q: Name a film that has won the Golden Bear in the Berlin Film Festival?
Q: Name a country that is developing a magnetic levitation railway system?
Q: Name the first private citizen to fly in space.

Q: What did Shostakovich write for Rostropovich?

Q: What is the term for the sum of all genetic material in a given organism?
Q: What is considered the costliest disaster the insurance industry has ever faced?
Q: What is Head Start?

Q: What was Agent Orange used for during the Vietnam War?

[2] Note that after the evaluation was officially over, we created new patterns in our system which cover some of these cases.

Q: What did John Hinckley do to impress Jodie Foster?

Q: What was the first Gilbert and Sullivan opera?

Q: What did Richard Feynman say upon hearing he would receive the Nobel Prize in Physics?
Q: How did Socrates die?

Q: Why are electric cars less efficient in the north-east than in California?

Figure 6: Harder questions in T200

We performed an analysis of the most frequent types of questions that occur in the evaluation corpus. Table 3 contrasts the performance of our system's best run on the different types of questions (represented as SYN-sets).

SYN-set	N	Score	Score/ N
PERSON NAME	30	16.5	55.0%
PLACE COUNTRY STATE NAME PLACEDEF	21	7.08	33.7%
NAME	18	3.67	20.4%
DATE YEAR	18	5.31	29.5%
PERSON ORG NAME ROLE	19	4.62	24.3%
undefined	19	11.45	60.3%
NUMBER	18	8.00	44.4%
PLACE NAME PLACEDEF	14	10.00	71.4%
PERSON ORG PLACE NAME PLACEDEF	10	3.03	30.3%
MONEY RATE	6	1.50	25%
ORG NAME	4	1.25	31.2%
SIZE1	4	2.50	62.5%
SIZE1 DURATION	3	0.83	27.7%
STATE	3	2.00	66.7%
COUNTRY	3	1.33	44.3%
YEAR	2	1.00	50.0%
RATE	2	1.50	75.0%
TIME DURATION	1	0.00	0.0%
SIZE1 SIZE2	1	0.00	0.0%
DURATION TIME	1	0.33	33.3%
DATE	1	0	0.00%

Table 3: Performance of A250 on different types of SYN-sets

8. RANKING SPANS

The two answer ranking systems, AnSel and Werlect use different algorithms which we describe in this section.

8.1 AnSel

The first algorithm that we used is called AnSel (ANswer SELection). It is essentially an optimization algorithm that uses 7 predictive variables to describe how likely a given span is to be the correct answer to a given question. The predictive variables are illustrated with examples related to the sample question number 10001 from TR38 "Who was Johnny Mathis' high school track coach?" and the potential answers to which are shown in Table 4.

Span	Type	Number	Rspanno	Count	Notinq	Type	Avgdst	Sscore	TOTAL
Ollie Matson	PERSON	3	3	6	2	1	12	0.02507	-7.53
Lou Vasquez	***PERSON***	***1***	***1***	***6***	***2***	***1***	***16***	***0.02507***	***-9.93***
Tim O'Donohue	PERSON	17	1	4	2	1	8	0.02257	-12.57
Athletic Director Dave Cowen	PERSON	23	6	4	4	1	11	0.02257	-15.87
Johnny Ceballos	PERSON	22	5	4	1	1	9	0.02257	-19.07
Civic Center Director Martin Durham	PERSON	13	1	2	5	1	16	0.02505	-19.36
Johnny Hodges	PERSON	25	2	4	1	1	15	0.02256	-25.22
Derric Evans	PERSON	33	4	4	2	1	14	0.02256	-25.37
NEWSWIRE Johnny Majors	PERSON	30	1	4	2	1	17	0.02256	-25.47
Woodbridge High School	ORG	18	2	4	1	2	6	0.02257	-28.37
Evan	PERSON	37	6	4	1	1	14	0.02256	-29.57
Gary Edwards	PERSON	38	7	4	2	1	17	0.02256	-30.87
O.J. Simpson	NAME	2	2	6	2	3	12	0.02507	-37.40
South Lake Tahoe	NAME	7	5	6	3	3	14	0.02507	-40.06
Washington High	NAME	10	6	6	1	3	18	0.02507	-49.80
Morgan	NAME	26	3	4	1	3	12	0.02256	-52.52
Tennesseefootball	NAME	31	2	4	1	3	15	0.02256	-56.27

Ellington	NAME	24	1	4	1	3	20	0.02256	-59.42
assistant	ROLE	21	4	4	1	4	8	0.02257	-62.77
the Volunteers	ROLE	34	5	4	2	4	14	0.02256	-71.17
Johnny Mathis	PERSON	4	4	6	-100	1	11	0.02507	-211.33
Mathis	NAME	14	2	2	-100	3	10	0.02505	-254.16
coach	ROLE	19	3	4	-100	4	4	0.02257	-259.67

Table 4: Feature set and span rankings for a sample question

8.1.1 Feature selection

The seven span features described below were found to correlate with the YES/NO categories of the training data. As an illustration, we use the answer to training question number 10001.

Number: position of the span among all spans returned. Example: "*Lou Vasquez*" was the first span returned by GuruQA on the sample question.

Rspanno: position of the span among all spans returned within the current passage.

Count: number of spans of any span class retrieved within the current passage.

Notinq: the number of words in the span that do not appear in the query. Example: **Notinq** ("*Woodbridge high school"*) = 1, because both "high" and "school" appear in the query while "Woodbridge" does not. It is set to –100 when the actual value is 0.

Type: the position of the span type in the list of potential span types. Example: **Type** ("*Lou Vasquez*") = 1, because the span type of "*Lou Vasquez*", namely "PERSON" appears first in the SYN-set, "PERSON ORG NAME ROLE".

Avgdst: the average distance in words between the beginning of the span and the words in the query that also appear in the passage. Example: given the passage *"Tim O'Donohue, Woodbridge High School's varsity baseball coach, resigned Monday and will be replaced by assistant Johnny Ceballos, Athletic Director Dave Cowen said."* and the span "*Tim O'Donohue*", the value of **avgdst** is equal to 8.

Sscore: passage relevance as computed by GuruQA.

8.1.2 AnSel Training Algorithm

The TOTAL score for a given potential answer is computed as a linear combination of the features described in the previous subsection:

$$\text{TOTAL} = \sum w_i f_i$$

The algorithm that the training component of AnSel uses to learn the weights used in the formula is shown in Figure 7.

For each <question,span> tuple in training set:

1. Compute features for each span
2. Compute TOTAL score for each span using current set of weights

Repeat

3. Compute performance on training set
4. Adjust weights w_i

Until performance > threshold

5. Store weights for use in ranking

Figure 7: Algorithm used by AnSel to learn the weights

8.1.3 AnSel Ranking Algorithm

Once AnSel has learned the weights during the training stage, it can be used to rank potential answers to other questions. The algorithm used is shown in Figure 8.

For each question in test set:

1. Compute features for each span
2. Compute TOTAL score for each span using weights w_i
3. Rank spans
4. Let current_span = highest ranked span
5. Let answer_set = {}

Repeat

6. Let current_extract = extract of 50 (or 250) bytes centered around current_span
7. Skip current extract if already included in answer_set
8. Insert current_extract in

```
    answer_set
        Until answer_set contains five extracts
9. Output answer_set
```

Figure 8: Algorithm used by AnSel for ranking potential answers

8.1.4 Example system run

For the question "Who was Johnny Mathis' high school track coach?", GuruQA retrieved a total of 23 spans (12 were tagged by Resporator as "PERSON", seven as "NAME", three as "ROLE", and one as "ORG"). The signature of the question indicates that these are the four possible span types for the answer, ordered PERSON, ORG, NAME, ROLE from the likeliest to the least likely.

The computed scores are based on the following scoring formula:

TOTAL (span) = – 0.3 * number – 0.5 * rspanno + 3.0 * count + 2.0 * notinq – 15.0 * types – 1.0 * avgdst + 1.5 * sscore

After AnSel has ranked all potential answers, it extracts a set of 50- and 250-byte passages that cover the top answers in the list. The actual extracts are shown in Figure 9 and Figure 10.

Document ID	Score	Extract
LA053189-0069	892.5	of O.J. Simpson , Ollie Matson and Johnny Mathis
LA053189-0069	890.1	Lou Vasquez , track coach of O.J. Simpson , Ollie
LA060889-0181	887.4	Tim O'Donohue , Woodbridge High School 's varsity
LA060889-0181	884.1	nny Ceballos , Athletic Director Dave Cowen said.
LA060889-0181	880.9	aced by assistant Johnny Ceballos , Athletic Direc

Figure 9: Fifty-byte extracts

Document ID	Score	Extract
LA053189-0069	892.5	Lou Vasquez , track coach of O.J. Simpson , Ollie Matson and Johnny Mathis during his 32-year career, died Saturday while at his South Lake Tahoe vacation cabin, it was announced Tuesday . He was 68 . His Washington High school teams won five consecu
LA060889-0181	887.4	Tim O'Donohue , Woodbridge High School 's varsity baseball coach , resigned Monday and will be replaced by assistant Johnny Ceballos , Athletic Director Dave Cowen said.
LA062090-0017	880.6	Civic Center Director Martin Durham said Mathis was to have entered the parking lot in a convertible Rolls-Royce to cut the ribbon for the dedication of Johnny Mathis Boulevard .
LA052390-0122	874.8	Ellington liked what he heard. Johnny Hodges was quitting the band and Morgan was invited to replace him, but he couldn't leave high school to go on the road. The new album's title track and " In a Sentimental Mood " are Morgan 's most recent homages
LA062389-0083	874.6	NEWSWIRE Johnny Majors , Tennessee football coach , said that prize recruit Derric Evans will not be allowed to play for the Volunteers because of his arrest Tuesday night in Dallas . Evans and a high school teammate, Gary Edwards , were charged with

Figure 10: Two-hundred-and-fifty-byte extracts

8.2 Werlect

The algorithm used to create VS50 and VS250 made use of many of the same features of noun phrases (spans), that were used in AnSel, but employed a different ranking scheme. We refer to this sister algorithm as Werlect (ansWER seLECTion).

8.2.1 Approach

Unlike AnSel, the algorithm developed for Werlect was based not on a simple linear function, but a two-step, rule-based process approximating a function that included interaction between variables. In the first stage of this algorithm, we assign a rank to every relevant noun phrase within each sentence according to how likely it is to be the target answer. Next, we generated and ranked each N-byte (N = 50 or 250) fragment, based on the sentence score given by Guru-QA, measures of the fragment's relevance, and the ranks of its component noun phrases. Werlect also differed from AnSel in its development in that there was no training algorithm, but was instead developed through manual trial-and-error, optimizing for the TR38 questions.

8.2.2 Step One: Feature selection

The first task in our two-step analysis is to rank each noun phrase, or span, for each hit returned by Guru-QA. In addition to the noun phrase's type, three main features were used to rank the noun phrases in each sentence. It is hypothesized that the target answer 1) is more likely to appear in multiple hits; 2) may contain, in some part, some of the query terms, and 3) is likely to be closer in proximity to matching query terms. With these rules, we hoped to identify the best noun phrase among several selected within one hit, and when necessary, promote a span higher than the rank awarded by Guru-QA.

Thus, the noun phrase features considered in Werlect are analogous to those used in AnSel, including **Type** (the position of the span type in the list of potential span types), **Avgdst** (the average distance in words between the beginning of the span and the words in the query that also appear in the passage), **Sscore** (passage relevance as computed by Textract and Resporator). Two additional features were also taken into account:

NotinqW: a modified version of **Notinq** (the number of words in the span that do not appear in the query). As in AnSel, spans that are contained in the query are given a rank of 0. However, partial matches are weighted favorably.

Frequency: how often the span occurs across different passages (not to be confused with AnSel's **Count**, which refers to the number of occurrences only within the current passage)

Examples of the isolated effects of Avgdst, NotinqW, and Frequency on answer selection are presented below.

Proximity (AveDist)

We hypothesize that the target answer is closer in proximity to the matched terms. Figure 11 shows a candidate answer that contains four noun phrases of the desired type, NUMBER. The noun phrases appear in bold, with their respective reciprocal average distances from the matched terms. The correct answer, 60 million, has the highest reciprocal average distance (.437) of the four noun phrases.

> "What is the number of buffaloes thought to have been living in North America when Columbus landed in 1492?"
>
> ". . . there are between 60,000 (.318) to 80,000 (.336) head of bison in America. . . . That's not many compared to the estimated 60 million (.437) that inhabited North America when Columbus discovered it in 1492, or even compared to the 20 million (.25) that still roamed the Great plains in the 1850's."

Figure 11: Text passage with four potential answers

This criterion effectively helps to identify one potential answer over the others within a single passage containing several candidates.

Relevance (NotinqW)

Although we know that a noun phrase that is completely contained in the query cannot be the answer, a noun phrase that contains part of the query may be more relevant. For example, if the question asks, "Who was Lincoln's Secretary of State?" a noun phrase that contains "Secretary of State" is more likely to be the answer than one that does not. In this example, the noun phrase, "Secretary of State William Seward" is the most likely candidate, based on this criterion.

This criterion may also play in a role in the event that Resporator fails to identify relevant phrase types. For example, in the training question, "What shape is a porpoise's tooth?" the phrase "spade-shaped" is chosen from among all nouns and adjectives of the sentences returned by Guru-QA.

Frequency

We hypothesize that a correct answer is more likely to occur in multiple hits than incorrect answers. For example, the test question, "How many lives were lost in the Pan Am crash in Lockerbie, Scotland?" resulted in four answers in the first two sentences returned by Guru-QA. Table 5: Influence of frequency on final 50-byte span rank shows the

frequencies of each term, and their eventual influence on the span rank.

Initial Sentence Rank	Noun Phrase	Frequency	Span Rank
1	Two	5	2
1	365 million	1	3
1	11	1	4
2	270	7	1

Table 5: Influence of frequency on final 50-byte span rank

Without considering the criterion described here, the competing noun phrases from a single answer, such as "two," "365 million," and "11," are tied, and are essentially arbitrarily ranked in first, second, and third place. Taking the frequency into consideration, the phrase "two" is awarded the top rank within that answer, yielding a span rank of 2 out of all possible 50-byte spans. However, the correct answer, "270," occurs seven times among the top ten answers returned by the search engine, serving to promote the fragment that spans it to first place.

8.2.3 Step two: Ranking the Sentence Spans

After each relevant noun phrase is assigned a rank, spans of 50 (or 250) bytes of all answers are created. Then, to each is assigned a score that is equal to the sum of the noun phrase ranks plus additional points for other words that match the query.

8.2.4 Algorithm

The algorithm used by Werlect is shown here.

For each question in test set:

1. Let noun_phrase_set = all potential answers
2. For each sentence, rank the candidate phrases
3. Extract all spans of 50 (or 250) bytes
4. Rank and sort all spans based on phrase ranks, matching terms, and sentence rank
5. For each noun_phrase in noun_phrase_set,

   REPEAT
   - Let highest_ranked_span = highest-ranked span containing noun_phrase
   - Let answer_set[i++] = highest_ranked_span
   - Remove every noun_phrase from noun_phrase_set that is found in highest_ranked_span
   - Exit if i > 5.
7. Output answer_set.

The entirety of this algorithm approximates the following function for the final rank:

$$Span\_Rank = f\left(\sum_{\#ofwords} noun\_phrase\_scores \sum_{\#ofwords} Avgdst / \#ofwords + Frequency + Sentence\_Rank\right.$$

where

Frequency_weight
= .1 if *frequency* < 3 and .2 if *frequency* >= 3

word_scores = f(Types, NotInQW)

This function is presented here for illustrative purposes only. Because the strategy in developing this algorithm did not involve optimization of this ranking function *per se*, a rigorous comparison with AnSel's ranking function would not be illustrative in this discussion.

9. EVALUATION

In this section, we describe the performance of our system on the training data and on the test data. For the test data, we refer to the four runs that we submitted officially. These four runs are labeled as follows: A50 (AnSel on 50 bytes), A250 (AnSel on 250 bytes), W50 (Werlect on 50 bytes), and W250 (Werlect on 250 bytes).

9.1 Evaluation scheme

For each question, the performance score is computed as the rank of the first correct answer given by the system and the reciprocal value of the rank (RAR). For example, if the system has given the correct answer in two positions: second and fifth, the score for that question will be ½. The same score would be obtained if the system gave the correct answer in only position 2.

To compute the overall performance of the system, we use the Mean Reciprocal Answer Rank (MRAR):

$$MRAR = 1/n \left(\sum_i 1/rank_i\right)$$

9.2 Performance on TR38

The system performance on TR38 is shown in Table 6. We were able to cover with the first anwers 14 questions out of 38 while answering 7 others within the first five answers.

	First	Second	Third	Fourth	Fifth	TOTAL
# cases	14	2	2	1	2	21
Points	14.00	1.00	0.67	0.25	0.40	16.32

Table 6: Performance on TR38

9.3 Performance on official evaluation data

The performance of A50 on T200is shown in Table 7. We achieved a total MRAR score of 63.22 which means that we got 49 questions among the 198 right from our first try and 39 others within the first five answers.

	First	Second	Third	Fourth	Fifth	TOTAL
# cases	49	15	11	9	4	21
Points	49.00	7.50	3.67	2.25	0.80	63.22

Table 7: Performance of A50 on T200

The performance of A20 on T200 is shown in Table 8. Here we were able to answer 71 questions with our first answer and 38 others within our first five answers. Overall our MRAR score is 85.17 (out of 198).

	First	Second	Third	Fourth	Fifth	TOTAL
# cases	71	16	11	6	5	21
Points	71.00	8.00	3.67	1.50	1.00	85.17

Table 8: Performance of A250 on T200

The next table shows some statistics on our two best runs (for 50 and 250 bytes, respectively)

To give a better idea of the performance of our system, we split the 198 questions into 20 groups of 10 questions (or 9 questions in the two cases in which the evaluators removed questions from the original 200-question set). Our performance on a group of questions ranged from 0.87 to 5.50 points for the 50-byte run (A50) and from 1.98 to 7.5 points for the 250-byte run (A250). The results are shown in Table 9.

	50 bytes	250 bytes
n	20	20
Avg	3.19	4.30
Min	0.87	1.98
Max	5.50	7.50
Std Dev	1.17	1.27

Table 9: Performance on groups of ten questions

The final evaluation (included in Table 10) shows how well our system did compared to the rest of the 25 participants in the TREC Q&A evaluation.

Run	Median Average	Our Average	# Times Our Run > Median	# Times Our Run = Median	# Times Our Run < Median
W50	0.12	0.28	56	126	16
A50	0.12	0.32	72	112	14
W250	0.29	0.39	60	106	32
A250	0.29	0.43	66	110	22

Table 10: Comparison of IBM's system and the other participants

10. CONCLUSION

We presented a new technique for finding answers to natural language questions using text corpora as reference.

We showed that a span-centered approach to question answering can deliver very good results.

We described seven features that correlate with the plausibility of a given text span being a good answer to a question. These features were described in detail in Section 8.1. We showed that a linear combination of these features performs well on the task of ranking text spans by the estimated relevance to the natural language question.

In the future, we plan to concentrate on getting better categories of text spans in order to provide fine-grained matches between question types and span types. We also intend to perform large-scale parameter learning.

We know we need to expand the set of question templates for existing QA-Tokens, as well as add more QA-Tokens and corresponding templates for a broader set of syntactic quantities.

Finally, we plan to investigate how language reuse and regeneration (LRR) [2] techniques can be used to provide contextual answers to natural language questions in a dialogue environment.

11. REFERENCES

[1] TREC Q&A Evaluation official Web site: http://www.research.att.com/~singhal/qa-track.html

[2] Dragomir R. Radev. "Language Reuse and Regeneration: Generating Natural Language Summaries from Multiple On-Line Sources", PhD thesis, Department of Computer Science, Columbia University, New York, October 1998.

[3] AAAI Fall Symposium on Question Answering, North Falmouth, MA, 1999.

[4] V.A. Kulyukin, K.J. Hammond, and R.D. Burke. "Answering Questions for an Organization Online", Proceedings of AAAI'98.

[5] Julian Kupiec "Murax: A Robust Linguistic Approach for Question Answering Using an On-line Encyclopaedia", Proceedings of SIGIR'93.

[6] Roy Byrd and Yael Ravin. "Identifying and Extracting Relations in Text", Proceedings of NLDB 99, Klagenfurt, Austria.

[7] Nina Wacholder and Yael Ravin and Misook Choi. "Disambiguation of Proper Names in Text", Proceedings of ANLP'97. Washington, DC, April 1997.

[8] Dragomir Radev and Kathleen McKeown. "Building a generation knowledge source using internet-accessible newswire". Proceedings of ANLP'97. Washington, DC, April 1997.

[9] George Miller. "WordNet: A Lexical Database for English", Communications of the ACM 38(11) pp 39-41, 1995.