MITRE's Qanda at TREC-12

John D. Burger The MITRE Corporation *john@mitre.org*

Introduction

Qanda is MITRE's TREC-style question answering system. This year, we were able to apply only a small effort to the TREC QA activity, approximately one person-month. As well as some general improvements in Qanda's processing, we made some simple attempts to handle definition and list answers.

1. TREC-12 system description

Catalyst architecture

Qanda uses a general architecture for human language technology called *Catalyst*, (Burger & Mardis 2002, Nyberg et al., to appear). Developed at MITRE for the DARPA TIDES program, the Catalyst architecture is specifically designed for fast processing and for combining the strengths of Information Retrieval and Natural Language Processing into a single framework. Catalyst uses a dataflow architecture in which standoff annotations are passed from one component to another, with the components connected in arbitrary (acyclic) topologies. The use of standoff annotation permits components to be optimized for just those pieces of information they require for their processing.

Major system components

Qanda has a by now familiar architecture—questions are analyzed for expected answer types, documents are retrieved using an IR system and are then processed by various taggers to find entities of the expected types in contexts that match the question. Below we describe each of the major components in turn.

- Question analysis: This component is run after the question has been subjected to POS and named entity tagging. It uses a simple grammar, currently hand-written, to identify important components of the question—see Section 2 below for more detail.
- IR wrappers: Catalyst components have been written for several IR engines, taking the results of the question analysis and formulating an IR query. For continue to use the Java-based Lucene engine (Apache 2002). Lucene's query language has a phrase operator, and also allows query components to be given explicit weights. Qanda uses both of these capabilities in constructing

queries from the information extracted from the question. For TREC-12, the top 25 documents were retrieved.

- Passage processing: Retrieved documents are tokenized, and sentence boundaries are detected. Because some downstream components run more slowly than the rest of the system, Qanda assigns a preliminary score to each sentence by summing the log-IDF (inverse document frequency) of each word that overlaps with the question. Those sentences with a low score are filtered out and not processed by most of the system.
- Named entity tagging: Qanda uses Phrag (Burger et al. 2002), an HMM-based tagger, to identify named persons, locations and organizations, as well as temporal expressions. Phrag is also used as a POS tagger for question analysis.
- Numeric tagging: A simple pattern-based tagger identifies measures, currency, percentages and other numeric phrases.
- Specialized taggers: We have a simple facility for constructing taggers from fixed word- and phraselists. These were used to re-tag many named locations more specifically as cities, states/provinces, and countries. Qanda also identifies various other (nearly) closed classes such as precious metals, birthstones, various animal categories, etc.
- Overlap: The question is compared to each sentence, and a number of overlap features are computed, some in terms of various WordNet relations (see Section 3).
- Answer formation and ranking: Candidates are identified and merged, a number of features are collected, and a score is computed (Section 3).

For factoid questions, we simply use the top-scored candidate. For definition and list questions, we apply some other processing to generate answer strings, as detailed in Section 4

2. Questions analysis

Phrag, our HMM-based tagger, first annotates questions using separate models for part-of-speech and named entities. Qanda also runs a simple lookup-based tagger that maps head words to answer types in Qanda's ontology using a set of approximately 6000 words and phrases, some extracted semi-automatically from WordNet, some identified by hand. Based on these annotations, Qanda's main question analysis component uses a parser with a simple hand-optimized grammar to identify the following aspects of each question:

- Answer type: a type in Qanda's (rather simple) ontology, e.g., PERSON or COUNTRY.
- Answer restriction: an open-domain phrase from the question that describes the entity being sought, e.g., first woman in space.
- Salient entity: What the question is "about". Typically a named entity, this corresponds roughly to the classical notion of topic, e.g., Matterhorn in What is the height of the Matterhorn?
- Geographical restriction: Any phrase that seems to restrict the question's geophysical domain, e.g., in America.
- *Temporal restriction*: Any phrase that similarly restricts the relevant time period, e.g., in the nineteenth century.

These features are emitted as annotations on the question, and are then available for other components to use.

3. Answer ranking with log-linear models

Qanda only examines sentences that match the question sufficiently, based on the IDF-weighted overlap described above. It collects candidate answers by gathering phrasal annotations from all of the semantic taggers, and identifies a number of features. These are combined using a conditional log-linear model trained from past TREC QA data sets. This approach has been used by several TREC participants, e.g., Ittycheriah et al (2001).

Many of the features used in the log-linear model reflect particular kinds of overlap between the question and the context in which the candidate answer is found:

• Context IDF Overlap: Described above.

- Context Unigram Overlap: Raw count of words¹ in common with the question.
- Context Bigram Overlap: Raw count of word bigrams in common with the question.
- Context Question Restriction Overlap: Raw count of words from the restriction phrase of the question (see Section 2).
- Context Salient Overlap: Raw count of words considered especially salient by question analysis (see Section 2).
- *Context Synonym Overlap*: Raw count of words that could be synonymous with questions words.
- *Context Antonym Overlap*: Raw count of words that could be antonyms of question words.

The synonym and antonym features are computed with respect to WordNet (Fellbaum 1998).

Several features are computed based on the candidate itself, or its location in the context sentence:

- Candidate Overlap: Raw count of words in common between the candidate itself and the question, to bias against entities from the question being chosen as answers.
- Candidate Overlap Distance: Number of characters between the candidate and the closest (content) question word in the context.²
- Candidate Question Restriction Distance: Number of characters between the candidate and a word from the restriction phrase of the question.

The only document-level feature currently used is the following:

• *IR Ranking* of the source document by the IR system.

Candidates with similar textual realizations are merged, with the combined candidate retaining the highest value for each feature. Accordingly, an additional feature is maintained:

• Merge Count

A number of boolean features are also computed that compare the question's expected answer type derived by analysis with the semantic type of the candidate:

¹All of the "raw count" features described in this section omit stop words.

²Words would arguably be a more intuitive unit for this feature.

- *Type Same*: Boolean, true if the candidate and expected answer types are identical.
- *Type Consistent*: True if the candidate's type is "similar" to the expected answer type.

For the most part, candidates are only considered for a question if their types are consistent. For example, Where questions lead to an expected answer type of *LOCATION*, which is consistent with *COUNTRY* candidates; *How much* questions lead to *QUANTITY*, consistent with *PERCENTAGE*.

Ideally, Qanda would consider all candidates for all questions, but, if nothing else, performance considerations justify limiting this. We do not even represent all consistent pairs as explicit features. Instead, a small set of approximately 20 combinations was chosen by hand, as indicated in Figure 1. These represent particular biases or preferences that we feel justified in trying to acquire from the training data. In addition, some of these pairwise features represent exceptions to the consistency requirement, e.g., PERSON is not consistent with COUNTRY, but we wish to consider such candidates anyway. Similarly, we wish to consider certain named entity types as candidates, even when question analysis was unsuccessful in divining an expected answer type (unknown).

Question expected answer type	Candidate
PERSON	type. ORGANIZATION
PERSON	COUNTRY
NAME	PERSON
NAME	ORGANIZATION
NAME	LOCATION
CITY	LOCATION
DATE	YEAR
DATE	YEAR
ORGANIZATION	other
AMBIGLONG	DURATION
AMBIGLONG	LENGTH
AMBIGBIG	LENGTH
AMBIGFAST	SPEED
MEASURE	MASS
MEASURE	MONEY
MEASURE	MISCMEASURE
MEASURE	other
QUANTITY	PERCENT
unknown	LOCATION
unknown	ORGANIZATION
unknown	PERSON

Figure 1: Type-pair features used in evaluating answer candidates

After all of the (merged) candidates have been acquired, the raw feature values described above are normalized with respect to the maximum across all candidates for a particular question, resulting in values between 0 and 1.³ Features normalized in this way are more commensurate across questions, especially word overlap and related features (Light et al. 2001).

The normalized features are then combined using the weights assigned by the log-linear model during training. This year, we trained the model using the questions from TREC 1999 and 2000. We also used a development test set composed of TREC 2002 factoid questions, the 25 AQUAINT definition evaluation questions, and the lists questions from 2001.

4. Special question types

This year, in addition to the usual factoid questions, systems had to deal with two additional types of questions, definition and list questions.

Definition questions

Qanda has no real facility for processing definition questions as such. Instead, we attempted to leverage our factoid question processing, which for the most part only considers named and other entities as candidate answers. Of course, very few definition answers were named entities, per se, but we noticed that certain kinds of named entities were involved with some definition answers, as indicated in the example below:

Who is Gunter Blobel?

Is at <u>Rockefeller University</u>
<u>1999</u> Nobel prize in Medicine
was born in <u>1936</u>
was born in Waltersdorf, Silesia, Germany

Qanda's question analysis component could already identify the semantic type of the definition target (e.g., *PERSON*, above). Since definition answers did not need to be exact, we decided to allow Qanda to consider certain entity types as "answers" to definition questions, deriving the actual answer string by extracting approximately 90 characters around the putative candidate.

We used the type-pair features described in Section 3 to license certain combinations of definition target type and candidate type, as shown in Figure 2.

³The normalized values are computed so that the intuitively "best" feature value is 1, the worst 0—this is primarily for the developers' convenience, but also so weights are all positive, and more easily reasoned about.

Definition	Candidate type.
target type	
PERSON	DATE
PERSON	YEAR
PERSON	PERSON
PERSON	LOCATION
PERSON	COUNTRY
PERSON	fragment
ORGANIZATION	LOCATION
ORGANIZATION	COUNTRY
ORGANIZATION	PERSON
ORGANIZATION	fragment
unknown	fragment

Figure 2: Type-pair features used in evaluating answer candidates

Additionally, we injected some non-entity candidates using crude heuristics for identifying short fragments occurring in appositional contexts. Our hope was that the type-pair features, as well as the candidate count feature, would allow the system to find some definition answers. As training data, we used 24 questions from TREC 1999 and 2000 that we determined were essentially definition questions.

We had some limited success with this approach. In the following examples, we indicate with underlining the actual "core" candidates that Qanda considered:

2112: Who is Antonio Coello Novello?

general, as <u>New York's</u> health commissioner on Thursday, drawing cries of betrayal from the abortion.

2385: What is the Kama Sutra?

the famous Indian manual of sexual knowledge, and his pursuit of Lalita is a kind of realization

These answers are hardly well-formed, but apparently relevant. It remains to be seen whether such a crude approach can be refined sufficiently to be useful, or if more sophisticated approaches are necessary.

List questions

Qanda currently treats list questions no differently from factoid questions, except that more than one answer is generated. We might have simply picked some fixed cutoff, say 3, and generated exactly three answers to every list question. We decided to attempt something slightly more sophisticated.

Since Qanda's candidate evaluation mechanism is probabilistic in nature, we decided to choose how many answers to generate dynamically, based on the expected value of the score we might receive. Each list question was to be scored using F-measure, the harmonic mean of precision and recall:

$$F = \frac{2PR}{(P+R)} = \frac{2c}{(n+r)}$$

Here, n is the number of answers we choose to generate, c is the number of correct answers we generate, and r is the total number of correct answers possible. We do not know in advance whether an answer is correct, but we can use Qanda's probabilistic score for the answers as the basis for an expectation of c. We have no real hope of estimating r, the number of correct answers possible, so we simply fixed this at the magic number 3.

Thus, our algorithm for generating list answers was to add each of the candidates to the answer set in turn, increasing n by one each time, and calculating the following expectation:

$$2 \prod_{i=1}^{n} s_{i}$$

$$E[F] \prod_{i=1}^{n} \frac{s_{i}}{(n+r)}$$

Here, s_i is the score assigned to candidate i. We stop generating candidates when this expectation begins to decrease. On this year's evaluation, this admittedly crude mechanism performed slightly better than simply generating one candidate per list question (F=0.07 vs. 0.05).

5. Conclusion

As well as the requisite description of this year's system architecture, we have discussed Qanda's question analysis and our use of log-linear models for answer selection. We intend to improve the latter with better features, feature combination, and more sophisticated models. We also presented some initial results with crude mechanisms for generating definition and list answers. We intend to improve these components substantially as well.

References

Apache Software Foundation, 2002. "Jakarta Lucene—Overview". http://jakarta.apache.org/lucene/.

John D. Burger, John C. Henderson, William T. Morgan, 2002. "Statistical named entity recognizer adaptation", *Proceedings of the Conference on Natural Language Learning*. Taipei.

John D. Burger, Scott Mardis, 2002. "Qanda and the Catalyst architecture", *AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*.

Christiane Fellbaum, ed. *WordNet: An Electronic Lexical Database*. MIT Press, 1998

Abraham Ittycheriah, Martin Franz, Salim Roukos, 2001. "IBM's statistical question answering system", *Proceedings of the Tenth Text Retrieval Conference (TREC-10)*.

Marc Light, Gideon S. Mann, Ellen Riloff, Eric Breck, 2001. "Analyses for elucidating current question answering technology", *Natural Language Engineering* 7(4).

Eric Nyberg, John D. Burger, Scott Mardis, David Ferrucci, to appear. "Software Architectures for Advanced Question Answering", in *New Directions in Question Answering*, ed. Mark Maybury. AAAI Press.