QA-LaSIE: A Natural Language Question Answering System

Sam Scott[1] and Robert Gaizauskas[2]

[1] sscott@ccs.carleton.ca
Institute for Interdisciplinary Studies
Carleton University, Ottawa, K1S 5B6, Canada
[2] r.gaizauskas@dcs.shef.ac.uk
Department of Computer Science
University of Sheffield, Sheffield, S1 4DP, UK

Abstract. QA-LaSIE was the heart of the University of Sheffield entry to the Question Answering track of TREC-9. By relaxing some of the strongest linguistic constraints, we achieved a very significant performance improvement over our TREC-8 system on both the TREC-8 and TREC-9 tasks. Whereas most systems returned answers that were always close to the maximum allowable length, our system was one of the only entries that tried to return an "exact answer" to a question.

1 Introduction

This paper describes a system to discover answers to questions posed in natural language against large text collections. The system was designed to participate in the Question Answering (QA) Track of the Text Retrieval Conferences (see `http://trec.nist.gov`) and therefore the definitions of "question" and "answer" that we adopt for this paper are those used in the TREC QA track (see section 2 below). While the system is a research prototype, it is clear that systems of this sort hold great potential as tools to enhance access to information in large text collections (e.g. the Web). Unlike a search engine, which returns a list of documents ranked by presumed relevance to a user query, leaving the user to read the associated documents to fulfil his information need, a question answering system aims to return the precise answer to a question leaving the user no further searching (though of course a link to the source document enables the user to confirm the answer).

The task of question answering should be of interest to the AI community for the simple reason that some form of Natural Language Processing must be used. There was not a single system entered in the TREC-9 QA Track that did not use some form of linguistic knowledge – no group took a purely statistical word counting approach.[1] Thus, unlike other tasks in Information Retrieval, question answering is one in which some form of NLP seems unavoidable. Nevertheless, our own experience over two years of participation shows that an overly strict or

[1] The least NLP to be found was in one of the IBM groups' submissions. But even this system used the WordNet hypernym hierarchy to gain an edge over a simple bag-of-words representation. Most systems used considerably more NLP.

formalistic approach may not succeed as well as one based on a mixture of formal NLP and ad-hoc heuristics. This year we achieved much better performance by relaxing some of the strict constraints we had employed for TREC-8.

The essence of our approach is to pass the question to an information retrieval (IR) system which uses it as a query to do passage retrieval against the text collection. The top ranked passages from the IR system are then passed to a modified information extraction (IE) system. Partial syntactic and semantic analysis of these passages, along with the question, is carried out to identify the "sought entity" from the question and to score potential matches for this sought entity in each of the retrieved passages. The five highest scoring matches become the system's response. It is our hypothesis that NLP techniques can contribute positively to QA capability.

2 The TREC Question Answering Track

The TREC-9 QA Track task was to return a ranked list of up to 5 answers to each of 693 previously unseen questions. The answers had to be passages from texts found in a (provided) 4GB newswire text collection. In TREC-9 there were two subtasks: 50-byte and 250-byte answers (maximum). The score assigned to each question was the reciprocal of the rank at which the first correct answer was found, or 0 if no answer was correct. So a system got 1 point for a correct answer at rank 1, 1/2 for rank 2, etc. The final score assigned to the system was the Mean Reciprocal Rank over the entire question set. For more details see the QA track guidelines document [5].

3 System Description

3.1 Overview

The key features of our system setup, as it processes a single question, are shown in Figure 1. First, the (indexed) TREC document collection is passed to an IR system which treats the question as a query and returns top ranked passages from the collection. As the IR system we use the Okapi system [6][2] to retrieve passages between 1 and 3 paragraphs in length – a configuration arrived at experimentally (details in [7]). Following the passage retrieval step, the top 20 ranked passages are run through a filter to remove certain formatting features which cause problems for downstream components. Finally, the question itself and the filtered top ranked passages are processed by a modified version of the LaSIE information extraction system [3], which we refer to below as QA-LaSIE. This yields a set of top ranked answers which are the system's overall output.

The reasoning behind this choice of architecture is straightforward. The IE system can perform detailed linguistic analysis, but is quite slow and cannot process the entire TREC collection for each query, or even realistically preprocess it in advance to allow for reasonable question answering performance

[2] Software available at: `http://dotty.is.city.ac.uk/okapi-pack/` .

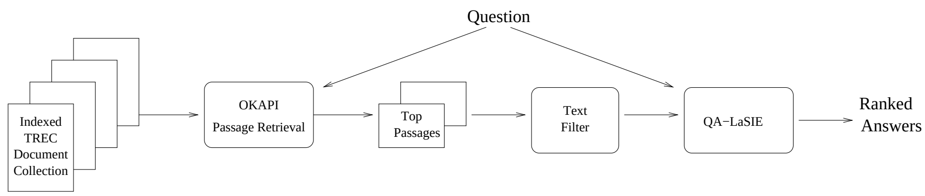

Fig. 1. System Setup for the Q & A Task

during the test run. IR systems on the other hand are designed to process huge amounts of data. By using an IR system as a filter to an IE system we hope to benefit from the respective strengths of each.

3.2 LaSIE

The system used to perform detailed question and passage analysis is largely unchanged in architecture from the LaSIE system entered in the last Message Understanding Conference (MUC-7) [3]. The system is essentially a pipeline consisting of the following modules, each of which processes the entire text[3] before the next is invoked.

Tokenizer. Identifies token boundaries and text section boundaries (text header, text body and any sections to be excluded from processing).

Gazetteer Lookup. Identifies single and multi-word matches against multiple domain specific full name (locations, organisations, etc.) and keyword (company designators, person first names, etc.) lists, and tags matching phrases with appropriate name categories.

Sentence Splitter. Identifies sentence boundaries in the text body.

Brill Tagger. Assigns one of the 48 Penn TreeBank part-of-speech tags to each token in the text [1].

Tagged Morph. Simple morphological analysis to identify the root form and inflectional suffix for tokens which have been tagged as noun or verb.

Parser. Performs two pass bottom-up chart parsing, pass one with a special named entity grammar, and pass two with a general phrasal grammar. A "best parse" is then selected (which may be only a partial parse) and a predicate-argument representation, or quasi-logical form (QLF), of each sentence is constructed compositionally.

Name Matcher. Matches variants of named entities across the text.

Discourse Interpreter. Adds the QLF representation to a semantic net, which encodes background world and domain knowledge as a hierarchy of concepts. Additional information inferred from the input using this background knowledge is added to the model, and coreference resolution is attempted between instances mentioned in the text, producing an updated discourse model.

[3] In the current implementation a "text" is either a single question or a candidate answer passage.

3.3 QA-LaSIE

The QA-LaSIE system takes a question and a set of passages delivered by the IR system and returns a ranked list of proposed answers for the question. The system is composed of the eight modules described in the preceding section plus one new module. Four key adaptations were made to move from the base IE system to a system capable of carrying out the QA task:

1. a specialised grammar was added to the parser to analyse questions;
2. the discourse interpreter was modified to allow the QLF representation of each question to be matched against the discourse model of a candidate answer passage;
3. the discourse interpreter was modifed to include an answer identification procedure which scores all discourse entities in each candidate passage as potential answers;
4. a TREC Question Answer module was added to examine the discourse entity scores across all passages, determine the top 5, and then output the appropriate answer text.

Parsing: Syntactic and Semantic Interpretation In the LaSIE approach, both candidate answer passages and questions are parsed using a unification-based feature structure grammar. The parser processes one sentence at a time and along with the original words of the sentence also receives as input a part-of-speech tag for each word, morphological information for each noun and verb (word root plus affix), and zero or more phrases tagged as named entities. As output the parser produces a representation of the sentence in a "quasi-logical form" – a predicate-argument representation that stands somewhere between the surface form of the sentence and a fully interpreted semantic representation in a standard logical language. In particular the QLF representation defers issues of quantifier scoping and of word sense disambiguation.

To take a simple example, the sentence fragment *Morris testified that he released the worm ...* is parsed and transduced to the representation

```
person(e1),name(e1,'Morris'),gender(e1,masc),testify(e2),
time(e2,past),aspect(e2,simple),voice(e2,active),lsubj(e2,e1),
release(e3),time(e3,past),aspect(e3,simple),voice(e3,active),
pronoun(e4,he),lsubj(e3,e4),worm(e5),number(e5,sing),
det(e5,the),lobj(e3,e5),proposition(e6),main_event(e6,e3),
lobj(e2,e6)
```

The name information is derived from the Gazetteer lookup stage (where *Morris* is recorded as a male first name), the tense information from the morphological analysis stage, and the grammatical role information from annotations on context-free rules in the grammar. In this case these rules encode that in English sentences which consist of a noun phrase followed by a verb phrase, which in turn consists of a verb in the active voice and a sentential complement, the noun phrase prior to the verb is the subject and the sentence following it is the object. For common nouns and verbs, the lexical root of the word becomes a predicate in the QLF language.

Both noun phrase heads and verb group heads are given unique discourse entity references of the form e_n. This allows modification relations (e.g. of prepositional phrases) or grammatical role information (e.g. subject and object relations) to be captured via binary predicates holding of these entities. In cases where parsing fails to capture all this information (e.g. when only simple noun phrase, verb group, prepositional phrase or relative clause chunks are found and not a spanning parse for the sentence) then partial QLF information can be returned, making the system robust in the face of grammatical incompleteness.

Each sentence in a candidate answer passage is analysed in this fashion. So is the question, using a special question grammar. This grammar produces a QLF for the question in much the same style as above. For example, a question such as *Who released the internet worm?* would be analysed as:

```
qvar(e1),qattr(e1,name),person(e1),release(e2),time(e2,past),
aspect(e2,simple),voice(e2,active),lsubj(e2,e1),
worm(e3), number(e3,sing),det(e3,the),lobj(e2,e3),
name(e4,'Internet'),qual(e3,e4)
```

Note the use of the special predicate, `qvar` (question variable), to indicate the "sought entity" requested by the question. In this case the `qvar` can also be typed because *who* tells us the entity of concern is a person, and we presume (by encoding this in the transduction rules) that the attribute we are seeking here is a name (and not, e.g., a definite description such as *a guy at MIT*). The fact that the system should return a name is encoded in the `qattr` predicate. In other cases where the interrogative pronoun is more generic (e.g. *what*) the type of the `qvar` and the attribute sought of it may not be so readily determinable.

Discourse Interpretation of Candidate Answer Passages Once a passage has been parsed and each sentence has been assigned a QLF representation, the discourse interpreter integrates the passages into a discourse model - a specialisation of a semantic net which supplies the system's background domain knowledge. For IE applications, this domain-specific background knowledge assists in extraction tasks by allowing template slot values to be inferred from it together with information supplied in the text being analyzed. However, for the TREC QA task there is no specific domain, and so this role of the semantic net is not relevant (though a very basic "generic" world model is employed).

The real function of the semantic net in the QA task is to provide a framework for integrating information from multiple sentences in the input. As the QLF representation of each sentence is received by the discourse interpreter, each entity is added as an instance node in the semantic net associated with its type node (the single unary predicate in which it occurs) – e.g. given `worm(e5)`, `e5` is linked to the `worm` node in the net, if it already exists, and to a new, dynamically-created node labelled `worm` if not. Added to each such entity node is an attribute-value structure, or property list, containing all the attribute and relational information for this entity (all binary predicates in which it occurs).

In addition to adding a sentence's QLF to the semantic net in this fashion, one further node is added representing the sentence itself. This sentence entity has a sequence number indicating the sentence's position in the passage, and also has an attribute recording the entity numbers of every entity occurring in the passage. Thus, the discourse model aims to model not only the content of the discourse, but simple aspects of the discourse structure itself.

After each sentence has been added to the discourse model, the discourse interpreter begins its main task – to determine coreference relations between entities in the current sentence and entities already added to the model from previous sentences in the input. There is not space to detail this algorithm here (see [2]), but in essence it relies upon factors including the semantic type compatibility, attribute compatibility, and textual proximity of potential coreferents. Once a coreference has been established between two entities, the two are merged by replacing all references to the two entity numbers by references to just one of them. However, the surface realisations which initially served as triggers for the creation of each distinct entity node are retained as attributes of the merged entity, and can be used later, e.g. to generate a text string as an answer.

Answer Identification After the discourse model has been constructed for a candidate answer passage, the QLF of the question is added to this model and treated as sentence 0. The coreference procedure is run and as many coreferences as possible are established between entities in the question and those in the passage[4].

In the TREC-8 version of QA-LaSIE [4] this procedure was the primary question answering mechanism: if the `qvar` was resolved with an entity in the candidate answer passage then this entity became the answer; if not, then no answer was proposed. This approach had several major drawbacks. First, it permitted only one answer per question, whereas the QA track allows up to five answers to be proposed. Second, it was very fragile, as coreference tends to be difficult to establish.

Given these weaknesses, the TREC-9 system follows a significantly different approach. Instead of attempting to directly corefer the `qvar` with an entity in the candidate answer passage, entities in the passage are scored in a way which attempts to value their likelihood as answers. The best scores are then used to select the answers to be returned from the passage.

The discourse model is transversed twice, sentence by sentence.

1. *Sentence Scoring* On the first pass, the sentences are given an integer score. The entities in the question are interpreted as "constraints" and each sentence in the answer passage gets one point for each constraint it contains. This rewards sentences for containing entities that have been detected as coreferring with entities in the question. Typically these will be sentences

[4] The standard coreference procedure uses a distance metric to prefer closer to more distant potential coreferences. Clearly this is irrelevant for questions which are not part of the original candidate answer passage. Hence we switch off the distance-preference heuristic for coreference in this case.

which contain named entities mentioned in the question, or sentences which contain definite noun phrases or pronouns which have already been resolved (as part of discourse interpretation of the passage). Sentences also get an extra point if they contain an event entity of the same type as the event derived from the matrix verb of the question (unless that verb is *to be*).

2. *Entity Scoring* On the second pass, the system looks in each sentence for the best possible answer entity. To be considered, an entity must be an object (not an event), and must not be one of the constraints from the previous step. If the `qvar` has a `qattr`, then the entity must also have the specified attribute to be considered a possible answer. The entities in a given sentence are compared to the `qvar` and scored for semantic similarity, property similarity, and for object and event relations.

   Semantic and property similarity scores are determined as for generic coreferencing. The semantic similarity score indicates how closely semantically related two things are (on a scale of 0 to 1). The semantic similarity is related to the inverse of the length of the path that links the two semantic types in the ontology. If the `qvar` and an entity have the same type (e.g. `person`), then that entity will receive a semantic similarity of 1. If the two semantic types are on different branches of the hierarchy, the score is 0.

   The property similarity score is also between 0 and 1 and is a measure of how many properties the two instances share in common and how similar the properties are.

   The object relation and event relation scores were motivated by failure analysis on the original system and were tuned through test runs. The object relation score adds 0.25 to an entity's score if it is related to a constraint within the sentence by apposition, a qualifying relationship, or with the prepositions *of* or *in*. So if the question was *Who was the leader of the teamsters?*, and a sentence contained the sequence ... *Jimmy Hoffa, Leader of the Teamsters,* ... then the entity corresponding to *Jimmy Hoffa* would get the object relation credit for being apposed to *Leader of the Teamsters*.

   The event relations score adds 0.5 to an entity's score if:

   (a) there is an event entity in the QLF of the question which is related to the `qvar` by a `lsubj` or `lobj` relation and is not a `be` event (i.e. derived from a copula construction), and

   (b) the entity being scored stands in the same relation (`lobj` or `lsubj`) to an event entity of the same type as the `qvar` does. So if the question was, *What was smoked by Sherlock Holmes?* and the answer sentence was *Sherlock Holmes smoked a pipe*, then the entity *a pipe* would get the event relations credit for being in the `lobj` relation to the verb *to smoke*.

   This represents a significant weakening of the requirement in our TREC-8 system that the `qvar` had to match with an entity in the answer passage which stood in the same relation to its main verb as the `qvar` did with the main verb in the question, as well the main verbs and other complements being compatible. Here a bonus is awarded if this the case; there it was mandatory.

Finally, the entity score is normalized to bring it into the range [0,1]. This is motivated by the idea that if two sentences have equal scores from step 1 above, the entity score should break the tie between the two, but should not increase their scores to be higher than a sentence that had a better score from step 1. Normalizing the score improved performance slightly in tests on the TREC-8 questions.

3. *The Total Score* For every sentence, the "best" answer entity is chosen according to the Entity Scoring described above. The sentence and entity scores are then added together and normalized by dividing by the number of entities in the question plus 1. The sentence instance is annotated to include the total score, the best entity (if one was found), and the "exact answer". The exact answer will be the `name` of the best entity if one was identified during parsing. Otherwise this property is not asserted.

Answer Output The answer output procedure gathers the total scores, as described in the preceding section, from each sentence in each of the passages analyzed by QA-LaSIE, sorts them into a single ranking, and outputs answers from the overall five highest scoring sentences.

We submitted four runs to the TREC-9 evaluation – two in the 50-byte category and two in the 250 category. These four runs are explained below:

shef50ea This is the exact answer run. If a high scoring sentence was annotated with a `trec9_exact_answer` attribute then this is assumed to be the answer. If there is no exact answer, then the code looks for a `trec9_answer_entity` and outputs the longest realization of that entity as the answer. If there is no answer entity, which can happen occasionally, then a default string is output. In all cases, the string is trimmed to 50 bytes if necessary, by trimming characters from the left hand side.

shef50 For this run, the system looks for the first occurrence of the `trec9_answer_entity` in the sentence and then outputs 50 bytes of the sentence centered around that entity. The 50-bytes will never go outside of the answer sentence (if the first occurrence is the first word, then the 50 bytes will be the first 50 bytes of the sentence, and so on). If the sentence is shorter than 50 bytes, then the full sentence is output as the answer. If there is no answer entity, the middle 50 bytes are output.

shef250 Same as shef50, but up to 250-bytes or the full sentence is output (whichever is shorter).

shef250p For this run, the answer for shef250 is computed, then the answer is padded to 250 bytes if necessary by adding characters from the file to both ends, going outside the confines of the sentence if necessary.

4 Results

4.1 Development Results (TREC-8)

Development results for the four run types described in the preceding section are shown in Table 1. `shef-trec8` refers to the official results obtained by our

System	Run	Mean Reciprocal Rank	Correct Answers	Rank in Class
shef-trec8	50	.081	N/A	15/17
okapi-baseline	50	.157	N/A	14/17 (hyp)
shef50ea	50	.329	89/164	4/17 (hyp)
shef50	50	.368	98/164	3/17 (hyp)
shef-trec8	250	.111	N/A	22/24
okapi-baseline	250	.395	N/A	11/24 (hyp)
shef250	250	.490	127/164	4/24 (hyp)
shef250p	250	.506	130/164	4/24 (hyp)

Table 1. Results on TREC-8 Questions. Rank hypothetical where marked.

TREC-8 system in TREC-8. `okapi-baseline` refers to a naive approach that simply used Okapi passage retrieval with a maximum passage length of one paragraph and then trimmed this paragraph to 50 or 250 bytes. This method led to Mean Reciprocal Rank scores of 0.157 for the 50 byte responses and .395 for the 250 byte responses. This totally naive approach would have placed 14th of 17 entrants in the TREC-8 50-byte system ranking and joint 11th of 24 in the 250-byte system ranking. In both cases these results were considerably higher than our own entries in TREC-8. Thus, we started with a sobering baseline to contend with. However, following development of the new approach described above in section 3.3 and numerous experiments with various parameter settings we arrived at the best development results presented in Table 1.

4.2 Final Evaluation Results (TREC-9)

Mean reciprocal rank scores for the four Sheffield TREC-9 runs are shown in Table 2, for both lenient and strict scorings.[5] We have also included our system's ranking over all the systems entered and the mean score for all systems entered. In all cases the performance of the Sheffield system is very close to the mean. We have also used the Perl patterns supplied by NIST for the TREC-9 results to score the submitted runs and an `okapi-baseline` system ourselves. These results are reported in the Auto column.

System	Run	Mean Reciprocal Rank			% Correct Answers in Top 5			Rank in Class
		Strict	Lenient	Auto	Strict	Lenient	Auto	
shef50ea	50	.159	.172	.171	23.6	25.7	25.8	28/35
shef50	50	.206	.217	.233	31.1	32.1	35.2	21/35
mean (of 35)	50	.220	.227		31.0	32.2		
okapi-baseline	50			.111			21.9	
shef250	250	.330	.343	.348	48.5	49.4	51.5	28/43
shef250p	250	.345	.357	.365	50.9	51.3	53.7	23/43
mean (of 43)	250	.351	.363		49.0	50.5		
okapi-baseline	250			.328			55.6	

Table 2. TREC-9 Results

[5] In strict scoring, an otherwise correct answer was marked as wrong if there was no support for it in the text from which it was extracted.

5 Discussion

The TREC-9 results also reported on the mean byte length of answers for each submitted run. Most participants gave as much text as was allowed, resulting in mean byte lengths of more than 45 bytes in the 50 byte category for all but a handful of systems. Our shef50ea run (the exact answer run) was one of the few that had a lower mean answer length – less than 10 bytes in fact. While we do not know yet what the mean byte length would have been for correct answers, we can report that our system had the highest score of the four systems that returned answers with an average length under 10 bytes.

At this point we do not have the information to allow us to apportion faults between Okapi and QA-LaSIE. In training on the TREC-8 questions Okapi was returning answer-containing passages for about 83% of the questions. On this basis the best QA-LaSIE mean reciprocal rank scores obtained in development were around .37 for the 50-byte runs and just over .50 for 250-byte runs, as presented above in Table 1.

Thus the TREC-9 test results represent a significant drop with respect to training results. Nevertheless, with respect to our best TREC-8 Mean Reciprocal Rank results (.081 for the 50-byte run, .111 for the 250-byte run), these figures represent a very significant improvement, especially given that the question set is significantly larger and the questions are "real", as opposed to what were artificially created back-formulations in many cases in TREC-8. And, they validate the central hypothesis of our TREC-9 work that we should abandon our previous rigid approach in which candidate answer entities either met constraints imposed by the question or did not, in favour of a looser approach which scored them in terms of various factors which suggested that they might be an answer. Finally, note that in both training and testing, for 250 as well as 50 byte answers, QA-LaSIE performed better than the Okapi baseline system, indicating that the NLP analysis is yielding increased value over a naive IR-only approach.

References

1. E. Brill. A simple rule-based part-of-speech tagger. In *Proc. of the Third Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy, 1992.
2. R. Gaizauskas and K. Humphreys. Quantitative Evaluation of Coreference Algorithms in an Information Extraction System. In S. Botley and T. McEnery, editors, *Discourse Anaphora and Anaphor Resolution*. John Benjamins, London, 2000.
3. K. Humphreys, R. Gaizauskas, S. Azzam, C Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. Description of the LaSIE-II system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.
4. K. Humphreys, R. Gaizauskas, M. Hepple, and M. Sanderson. University of Sheffield TREC-8 Q & A System. In *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC 8)*, 1999.
5. TREC-9 Question Answering Track Guidelines. http://trec.nist.gov/act_part/-guidelines/QA_guidelines.html, 2000.
6. S. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC 8)*, 1999.
7. S. Scott and R. Gaizauskas. University of Sheffield TREC-9 Q & A System. In *Proceedings of The Ninth Text REtrieval Conference (TREC 9)*, 2000. To appear.