The University of Sheffield's TREC 2003 Q&A Experiments

Robert Gaizauskas, Mark A. Greenwood, Mark Hepple,
Ian Roberts, Horacio Saggion and Matthew Sargaison

`r.gaizauskas,m.greenwood,` } `@dcs.shef.ac.uk`
`m.hepple,i.roberts,saggion`

Department of Computer Science
University of Sheffield

1 Introduction

The systems entered by the University of Sheffield in the question answering track of previous TRECs have been developments of the system first entered in TREC 8 (Humphreys et al., 1999). Although a range of improvements have been made to the system over the last four years (Scott and Gaizauskas, 2000; Greenwood et al., 2002), none has resulted in a significant performance increase. For this reason it was decided to approach the TREC 2003 evaluation more as a learning experience than as a forum in which to promote a particular approach to QA. We view this as the beginning of a process that will lead to much fuller appreciation of how to build more effective QA systems.

Our efforts this year were focussed on a number of objectives:

1. to understand better how certain key components of our system architecture, specifically the information retrieval (IR) component that feeds the answer extraction component, were performing and how modifying their behaviour might affect overall system performance;

2. to implement a simple baseline system that would allow for comparison with the more linguistically motivated system we have entered in the past;

3. to build a dedicated subsystem for the definition question subtask that was introduced this year;

4. to preprocess the entire AQUAINT corpus to extract useful information for use by a QA system, in particular named entities in specific classes.

In the following section we first briefly summarise our overall system architecture and approach to QA – largely unchanged from previous years – and then discuss our work with respect to each of the above four objectives. With this detail as background, we then describe the three runs we submitted and review their results. While the final scores obtained are not high, they do not reflect the value of what was learned in the course of preparing for the evaluation, most of which was learned too late to feed through into in the system we entered.

2 System Development and Experimentation

As detailed in our previous years' TREC submissions, our core QA system consists of an IR system coupled to a natural language analysis system. The essence of the approach is to pass the question unmodified to the information retrieval (IR) system which uses it as a query to do passage retrieval against the text collection. The top ranked passages output from the IR system are then passed to a modified information extraction (IE) system. This system first carries out partial, robust syntactic and semantic analysis of these passages and of the question (in which a specific *"sought entity"* is determined), transducing them both into a predicate-argument or quasi-logical form (QLF) representation. In this representation the predicates are, for the most part, either the unary predicates formed from the morphological roots of nominal or verbal forms in the text or binary predicates from a closed set of grammatical relations (e.g. `object`, `subject`) or of prepositions (e.g. `in`, `after`).

Given these sentence level "semantic" representations of candidate answer-bearing passages and of the question, a discourse interpretation step then creates a discourse model of each retrieved passage by running a coreference algorithm against the semantic representation of successive sentences in the passage, in order to unify them with the discourse model built for the passage so far. This results in multiple references to the same entity across the passage being merged into a single unified instance. Next, coreference is computed again between the QLF of the question and the discourse model of the passage, in order to unify common references.

In these passage+question models, possible answer entities are identified and scored as follows. First each sentence in each passage is given a score based on counting matches of entity types (unary predicates) between the sentence QLF and the question QLF (similar to counting noun and verb overlap in word-overlap approaches). Next each entity from a passage not so matched with an entity in the question (and hence remaining a possible answer) gets a preliminary score according to (1) its semantic proximity (in Wordnet) to the type of the entity sought by the question and (2) whether or not it stands in a relation R to some other entity in the sentence in which it occurs which is itself matched with an entity in the question which stands in relation R to the sought entity (e.g. an entity in a candidate answer passage which is the subject of a verb that matches a verb in the question whose subject is the sought entity will have its score boosted). An overall score is computed for each entity as a function of its preliminary score and the score of the sentence in which it occurs.

Finally, the ranked entity list is post-processed to merge and boost the scores of multiple occurrences of the same answer found in multiple passages and the top scoring answer is then proposed as the answer to the question.

Overall the intention is that the matching of candidate answer entities to the sought entity of the question be guided primarily by semantic type similarity (so "who" questions should have persons proposed as answers), then by lexeme overlap between question and answer-bearing sentence, and finally by sharing of grammatical relations where they can be identified. Redundancy of the answer across the candidate answer bearing passages (and optionally across external resources such as the Web) is also taken into account.

2.1 Coupling IR and QA

Using an IR system as the first component in a QA system to retrieve relevant candidate answer-bearing passages is an approach widely adopted by TREC participants. It makes sense as a way of narrowing the collection size down to something manageable for the more detailed, and processor intensive, analysis required for answer extraction. However, in such an architecture the performance of the IR component cleary bounds the performance of the overall system.

For the past two TREC QA evaluations in which we have participated we have used the Okapi system (Robertson and Walker, 1999), a probabilistic IR system which has been evaluated very favourably in the TREC mainstream IR evaluations. Despite these favourable evaluations, our analysis of its performance in the context of QA has shown up a serious problem. Experiments detailed in (Roberts and Gaizauskas, 2004) suggest that when using the top 20 relevant passages, answer bearing passages are found for only 66% of the questions. This figure increases to about 80% if the top 100 passages are considered and only to 82% for the top 200 passages. Similar results have been found by Hovy et al (2000), who report that this figure rises only to 92% considering the top 1000 documents returned per question by their IR system.

Document Rank	*Coverage*	*Percentage Correct*
0	0.0%	0.000%
1	18.6%	11.549%
2	25.4%	14.930%
5	37.0%	16.056%
10	46.0%	17.746%
20	54.0%	18.592%
30	57.4%	16.620%
50	61.4%	17.183%
100	66.8%	16.056%
150	68.0%	13.803%
200	69.2%	12.676%

Table 1: Coverage and end-to-end performance when using Okapi.

Since in the past we had been considering only the top 20 passages per question, we were placing an upper bound of 60% of questions being answered correctly on our overall system before answer extraction even began. To address this major issue we pursued two lines of enquiry this year:

1. How does supplying more documents from lower down the IR system ranking affect overall QA performance?
2. Can better IR performance, as viewed from a QA perspective, and better combined IR/QA performance be obtained by adopting a more hands-on approach to IR, e.g. implementing an inverted index for boolean retrieval, and experimenting with query formulation from natural language questions?

2.1.1 Descending the Ranking

Experiments were carried out with the aim of finding if the increase in coverage (the percentage of questions for which at least one answer bearing document is returned) obtained by considering more documents improves the end-to-end performance of the question answering system. The top 200 relevant passages were retrieved from AQUAINT for 350 of the main track questions from TREC 2002[1]. Coverage and end-to-end performance was then determined at a number of document ranks giving the results shown in Table 1.

It should be clear from the results in Table 1 that although the best coverage is achieved by considering

[1] All of these questions were known to have at least one correct answer within the collection.

the top 200 relevant passages this does not translate to better end-to-end performance of the system as this is clearly achieved using only the top 20 relevant passages.

Although these results show that using Okapi as the IR engine yields a relatively low coverage and end-to-end performance it at least shows that the best results are obtained when using just twenty relevant passages. On the one hand these results are pleasing as this is the configuration we were and will continue to use, now with some justification. On the other hand, since this places an upper bound of 60% of less on the correctness of our QA system, it reveals that we must either change the approach to IR, to get more answer-bearing passages in the top ranks, or reduce the answer extraction component's tendency to get distracted by noise as it descends the ranking.

2.1.2 Boolean IR with MadCow

To experiment with a more hands-on approach to IR we implemented a boolean search engine called MadCow which does word indexing at the sentence level. That is, each document in the AQUAINT collection is sentence-split and an inverted index of all words, minus those in a stoplist, is created which associates with each word a list of its document plus sentence offset occurrences and a count of how many documents it occurs in across the collection (so-called document frequency). A basic boolean query language was implemented which supports queries of arbitrarily deeply nested conjunctions and disjunctions of search terms (words); negation is not supported at this point as there has not appeared to be a need for it. When a query is issued all sentences which satisfy the query are returned. Note that this means that, e.g., a document or even two adjacent sentences which satisfy the query will not be returned: the query must be satisfied by a single sentence. This constraint may prove to be too severe, but it is a principled starting point. Relaxing the matching conditions can straightforwardly be done in further work.

Given the availability of a boolean search engine, how can questions best be mapped into queries to maximise the number of answer-bearing sentences returned from the collection? Where a conventional ranked retrieval engine is used in a QA system, for example engines using a probabilistic model (like Okapi) or a vector-space model, a question can be used directly as an IR query to retrieve the documents that are to be analysed for possible answers. More complicated schemes can be employed, of course, but this simple approach is both possible and widely used.

For boolean retrieval engines such as MadCow, however, the task of formulating queries to retrieve documents relevant to answering some question is non-trivial. A query formed as a conjunction of the words in the question (omitting stoplist words, perhaps) will commonly be too restrictive, returning few or no documents, whereas a query that is a disjunction of the same words will commonly retrieve too many. The best approach for formulating boolean retrieval queries is an open research topic. In what follows, we will describe the particular query formulation approach we implemented.

This query formulation approach places particular importance on the recognition of names within questions, and knowledge of variant forms for names of the same person/company, as acquired from the corpus by the methods described in Sec. 2.4.1. Query formulation operates in two phases, of which the first phase works with only names recognised within questions, whilst the second phase uses other, non-name, words from the question. A preprocessing component is applied to each question, returning the names identified, plus any known variants of these names. These alternate forms are used to create a 'strong' boolean condition, which succeeds if all the words of any one of the name variants are found in a document. A 'weak' condition is also computed for each name, which is usually just the final word of the name. For example, the system might recognise the name *Bill Clinton* and suggest the variant *President Clinton*, giving rise to the strong condition: `(( Bill & Clinton ) | ( President & Clinton ))`, with the weak condition being just `( Clinton )`. A range of search expressions are generated for each question by conjoining over conditions from the identified names, where each name may be represented by either its strong or weak condition, or might be omitted. For example, for two names, we will have conditions such as $(str_1 \& str_2)$ and $(wk_1 \& str_2)$, but also just (str_1) and (wk_2) (giving eight in total, as the entirely null condition is excluded). Retrieval is done for all of these search expressions, whose 'specificity' is reflected in the size of the passage set returned, i.e. the less passages retrieved, the more specific the query. Working through these results in order of decreasing specificity, the system collects the result passages together (deleting duplicates) until either all results sets have been included or adding passages for the next set would make the collection exceed a specified upper limit of size (we used a limit of 250 in these experiments). At the end phase 1, the overall process may terminate if the collected passages number more than some specified lower limit of size (we used a limit of 100 in these experiments). If not, phase 2 is initiated.

If phase 1 ended because the passage set for some name condition `N` is too large, then phase 2 serves to elaborate this condition by constructing search conditions `W` built from non-name words in the question, i.e. constraints `(N & W)` are used to documents for addition to the collection. Otherwise, conditions `W` built from non-name words in the question are used on their own. The conditions `W` are built by taking non-stoplist words from the question, which are known to exist somewhere in the corpus (i.e. have document frequency > 0). For each such word, its known variant forms are accessed (i.e. inflectional variants, e.g. *decided/decide*, and related nouns/verbs, e.g. *decision/decide*) are disjoined together, and then the expressions that result for the different words in the question are conjoined together. If the search condition that results fails to retrieve enough documents, it is weakened by deleting the sub-expression for the question word having the highest document frequency in the corpus (which is thereby deemed to be the 'least informative' term). This process iterates until either enough documents have been collected, or the condition cannot be further weakened (i.e. when it is derived from only a single word of the question).

At the end of the two phases, the system has a collection of documents from which a maximum of 50 are returned, with a preference being given firstly to those extracted in phase 1, and secondly those having greatest overlap with the question. If the system's collected document set is empty (which might happen if all searches have retrieved either no documents or more than the specified upper limit), then the system will fall back to just using 50 documents from some oversized set produced at an earlier stage.

Clearly there are many ways in whic this particular query formulation approach might be varied or refined and empirical work is required for the further development of the approach.

2.2 A Simple Baseline System

For TREC-8 and 9, which required 50 or 250 byte answers, we were able to implement a simple baseline using Okapi only in which the central 50 or 250 bytes in the top-ranked passages were returned as an answer. This is not a sensible baseline for returning single, exact answers, however, so for the past few years we have not had a baseline system against which we could compare our more complex, linguistically motivated system.

To rectify this situation we decided to implement a simple system that works as follows. First, the approach starts with the limiting assumption that all questions can be answered by one or more entities from a fixed set of semantic types. The entity types which can be recognised include the standard named entity categories from the MUC competitions (persons, organizations, locations, times and dates, monetary amounts) plus a wider class of measures (space, mass, duration, etc.) and types frequently seen in previous TRECs, as discussed below in 2.5.1 Clearly this assumption is not always warranted as in the question *"How did Patsy Cline die?"* for which the correct answer is *"in a plane crash"* which is an event type and not an entity.

The operation of this simple baseline system is as follows:

- Question Typing: The expected answer type is determined using a set of hand coded rules which operate over the words in the sentence. For example a question containing the word "who" suggests that the answer will be of type Person.

- Information Retrieval: Assuming the question can be typed then the IR approach outlined in Section 2.1.1 is used to find the top 20 most relevant passages; if the question cannot be typed, then the system simply returns no answer for this question.

- Answer Extraction: All named entities of the correct type are extracted from the relevant documents and retained as possible answers unless they fail one of the following conditions:
  - The document from which the current entity is drawn must contain all the named entities found in the question, and
  - The current entity must have no overlap with the question.

The remaining entities are then grouped together using the following equivalence test (Brill et al., 2001): *two answers are said to be equivalent if all of the non-stopwords in one are present in the other or vice versa*. The most frequently occurring answer group is then proposed as the answer to the question or if the question requires multiple answers then all answers located are proposed in order of frequency of occurrence[2].

2.3 Answering Definition Questions Through Query Expansion

Definition questions require a different approach to that used by QA-LaSIE for answering factoid questions mainly because the questions contain very little information useful for finding definition-bearing documents. For example there are 1108 sentences in the AQUAINT collection which contain the word *"aspirin"* most of which do not include a definition. However, if we can determine from external sources that *"analgesic"* occurs frequently in sentences defining aspirin

[2] The longest realisation of the answer within the group is used along with the highest ranked document identifier, which can, on occasions, lead to unsupported answers – 15 of the 450 questions in this evaluation.

then we can reduce the search space to only eight sentences in AQUAINT. The following method is used to determine the secondary terms to help locate definition bearing sentences:

- Definiendum Extraction: As the definiendum can be a complex linguistic unit we rely on a chart parser to produce a syntactic representation of the question. The definiendum is then assumed to be the right most noun phrase (note we assume that all definiendums will be nouns).

- Pattern Generation: We generate definition phrases, such as *"aspirin is a"* and *"such as aspirin"* from a set of fifty "seed" patterns ready for later processing.

- Secondary Term Extraction: We rely on three external sources for secondary term extraction:

  - WordNet (Miller, 1995): All adjectives, nouns and verbs found in the glosses of the definiendum are extracted as are the related hypernyms of the definiendum.
  - Britannica: All pages containing the definiendum are retrieved and the nouns, verbs and adjectives found in sentences containing the definiendum are extracted.
  - Web: The definition phrases generated earlier are used to locate unique relevant documents on the web. The nouns, verbs and adjectives are then extracted from the sentences within these documents which contain the definiendum.

  A combined list of secondary terms, up to a maximum of n elements, is then generated as follows:

  - all secondary terms found in WordNet (m terms, $m \leq n$),
  - a maximum of $(n-m)/2$ terms from Britannica with a frequency greater than one,
  - web terms with a frequency greater than one are then appended to the list until the maximum of n is reached.

  The order of the list reflects a degree of confidence we have in the source and also the fact that the more a term is used in a particular "definition" context the more we believe it is associated with the definiendum.

- Query Generation and Passage Retrieval: We have tried two different approaches to passage retrieval which necessitate different approaches to query generation. The approaches are as follows:

  - When using the probabilistic IR engine, Okapi, the search query is composed of the tokens in the question and the full list of secondary terms found in the previous stage. The twenty most relevant passages within AQUAINT are then retrieved using this query.
  - In the case of the MadCow boolean search engine an iterative procedure is used to generate the search query. Suppose the term sought is composed of tokens $\{m_i\}$ ($1 \leq i \leq k$) and the list of secondary terms consists of tokens $\{s_j\}$ ($0 \leq j \leq l$), then during iteration p the following boolean search is tried (term conjunction is represented by & and term disjunction by $|$):

    $$(m_1 \& m_2 \& ... \& m_k \& (s_1|s_2|...|s_p))$$

    If the number of passages returned in iteration p is greater than 20, then the searching procedure stops. This procedure on the one hand limits the scope of the main term at each step by conjoining it with a secondary term and on the other hand broadens its scope by disjoining it with additional secondary terms. In a sense this approach is similar to the narrowing and broadening techniques used in the MURAX system (Kupiec, 1993). In our approach, the order in which the secondary terms are used to expand the query reflects their association with the main term; thus, this search procedure is expected to retrieve good passages.

- Definition Extraction: More than one definition can be extracted at this stage not only because different fragments cover different aspects of the definition (e.g. *"aspirin is a drug"* vs. *"aspirin is a blood thinner"*) but also because the definiendum can be ambiguous (i.e. there are seven senses of "battery" in WordNet). We restrict our analysis of definitions to the sentence level, as a sentence is considered definition-bearing if it matches one of the previously generated patterns or if it contains the definiendum and at least three secondary terms (this threshold was arrived at after several experiments on a small training set).

  Instead of returning the full sentence as a definition we return the suffix which contains the definiendum and all secondary terms appearing in the sentence. This is in a crude attempt to remove some of the unnecessary information the sentence may contain.

  In an attempt to return the same definition only once, a vector representation of the definition, consisting of its terms and term frequencies, is created. If the current definition is too similar to any of the previously extracted defintions then it is discarded. Similarity here is quantified as the cosine between two vector representations, i.e. two vectors v_1 and v_2 are considered similar if $cosine(v_1, v_2) > threshold$ (the $threshold$ was determined through experimentation over a small training set).

2.4 Corpus Pre-Processing

In previous years QA tracks we brought language processing modules to the QA task that were not tailored in any particular way to the evaluation corpus. Clearly it makes sense to ensure your tools are able to work effectively on the evaluation corpus and so to address this issue we carried out two activities, one to preprocess the corpus to attempt to extract information about all named entities in the corpus, the other to extend our POS tagger lexicon by attempting to assign tags to all words in the corpus not in the lexicon.

2.4.1 Automatically Acquiring Named Entity Information from AQUAINT

All groupings of two or more consecutive capitalised word (plus full-stops and the lower-case word ‘of’) were extracted from the corpus. This list, comprising 14,919,780 multi-word expressions in 2,615,767 distinct forms, ranging from “UnitedStates” to “Prime Minister Benjamin Netanyahu of Israel”, was used in developing an experimental automated Named Entity catalog.

Because of the size of the corpus and the fact that these groups are lacking in complex grammar, the process used was one of simple pattern matching against key-word list including Titles (‘Mr’, ‘President’, etc), Company markers (‘Ltd’, ‘AG’, etc), Place markers (‘Road’, ‘Bridge’, etc), Country and City names (‘Israel’, ‘New York’) and common First names (‘Benjamin’). This initial process identified nearly 800,000 distinct people names, 160,000 institution names and around 75,000 company names. Random sample analysis of the results suggest that the name identification precision and recall are in the region of 0.92 and 0.88 respectively.

A second phase in the process was to automatically group together similarly named people into equivalence classes. On the basis of probable gender, nickname information (e.g. Bill⟺William), middle-initial (where present), title similarity (e.g. Gen⟺General, but not Actress⟺Congressman) and simple spelling mistakes/deviations in long names (‘Gennady Zyuganov’⟺‘Gennadi Zyugavov’) the 800,000 people names were clustered down to 674,000 distinct individuals. 25 distinct instances of Bill Clinton’s name were found to be equivalent, ranging from “Billy Boy Clinton” to “William Milhous Clinton”, with the titles “President”, “Governor”, “Gov” and (incorrectly) “Senator”.

One beneficial side affect of this approach was the amount of ‘extra’ information that was gathered about individuals on the basis of frequency of associated title and country keywords. Without further analysis of the corpus, the list of named people already contains the correct country of origin and main title/profession of at least 30,000 members.

The ultimate aim in this work is to arrive at an improved indexing of the corpus in which every distinct variation on a person or company name is recognised as the correct unique equivalence class. For this year’s TREC, however, the process was not fully developed and so only a simple query expansion was implented for the MadCow system, as described in Sec. 2.1.2.

2.4.2 Lexicon Improvement for Part-of-Speech Tagging

During development we noticed that there were a large number of word types (over 180,000) found in the AQUAINT corpus which were not in the lexicon used by our part-of-speech tagger. As a majority of the further processing relies, at least in part, on the POS tags assigned to the words any improvement should be beneficial to the system as a whole as well as improving the tagger for use in other projects.

The approach taken to deal with these unknown words was as follows:

- The entire AQUAINT corpus was tokenized and sentence boundaries were determined.
- The tagger was adapted to tag unknown words with UNK instead of a default tag, and then the tagger was run over the sentence split corpus.
- All non-upper case words which were tagged as UNK were then extracted from the corpus along with their frequency of occurance.
- The POS tag for these unknown words was determined from the tag assigned in the British National Corpus (BNC). Unfortunately the BNC uses a different tag set to the tagger and so the tags had to be mapped to PTB tags using a mapping derived by Steve Abney (1997).

The above approach provides tags for 63633 of the 183648 tokens previously assigned UNK and doubles the size of the lexicon used by the POS tagger.

2.5 Miscellaneous Further Improvements

2.5.1 Improved Named Entity Recognition

Examination of questions used in the previous TREC question answering evaluations, as well as questions from various freely available sources on the web, showed that the answers to some questions can be drawn from a closed set of possibilities. For example the question ”What is the state flower of Hawaii?” should have a flower as the answer, however, we can reduce this to the set of known state flowers (which is finite) increasing the chance that a question answering system will choose the right flower as answer. To this end numerous gazetteer lists of (almost) closed sets of entities were built including: languages, birthstones, Greek, Roman and Egyptian Gods, national anthems, planets, spacecraft, and state flowers, birds, mottos, trees and nicknames

Further examination of our existing named entity recogniser showed that it provided very little support for recognising measurements (especially compound measurements such as speed - meters per second). This was rectified and the system can now recognise measurements of distance, mass, time, speed, temperature and currency plus combinations of these (e.g. 6 foot 6 inches, ten meters per second).

Combining these improvements with those of the previous section results in named entity recognition that is better suited to answering questions especially via the AQUAINT corpus.

2.5.2 Updates to QA-LaSIE

The only major change from the version of QA-LaSIE described in (Greenwood et al., 2002) was the way in which questions requiring multiple answers are handled. The list questions asked in TREC 2002 explicitly stated the number of answers expected whereas the questions used for this evaluation do not. As QA-LaSIE draws its answers from two locations, AQUAINT and the web, using Google, it simply returns the overlap between the two sets of answers. Unfortunately this can lead to the situation in which no answers are returned for a question, but the guidelines for the evaluation state that at least one answer must be given for each list question. In these instances we return the string UNKNOWN ANSWER so as to abide by the guidelines while still allowing us to easily analyse the output from the system.

3 Final Evaluation Results

From the development and experimentation detailed above, we configured three evaluation runs as follows.

shef12okapi This run consisted of using Okapi to retrieve the relevant documents (Section 2.1.1) and then using either QA-LaSIE, with the extended named entity transducer and gazetteer lists, (Sections 2.4.1 to 2.5.2) or the definition system (Section 2.3) to extract the possible answers.

shef12madcow This run is identical to shef12okapi other than the MadCow boolean search engine (Section 2.1.2) was used to retrieve the relevant documents instead of Okapi.

shef12simple This run consisted of using Okapi to retrieve the relevant documents (Section 2.1.1) and then using either the simple baseline system (Section 2.2) or the definition system (Section 2.3) to locate the possible answers.

The results from these three runs can be seen in Table 2. We discuss aspects of each in turn.

3.1 shef12okapi

Of the 413 factoid questions Okapi was able to find answer bearing passages[3] for 198 of them. As 30 of the questions have NIL as the correct answer then the system should be able to answer these as well giving a maximum attainable score of 0.552 (228/413). Unfortunately the official score for this run was only 0.046 (19/413). Part of the reason for such a low score is the way in which the system selects the answer text, always favouring the longest realisation, which causes quite a few correct answers to be classed as inexact – in this case 27 answer were marked as inexact.

For definition questions the system retrieved response sets from the AQUAINT collection for 28 of the 50 definition questions. Of these 28 questions, 22 contained at least one definition nugget all of which contained at least one essential nugget giving a score of 0.230.

3.2 shef12madcow

Of the 413 factoid questions MadCow was able to find answer bearing passages for 173 of them. As 30 of the questions have NIL as the correct answer then the system should be able to answer these as well giving a maximum attainable score of 0.492 (203/413). Unfortunately the official score for this run was only 0.063 (26/413). This run suffers the same problem as shef12okapi with 32 answers being marked as inexact.

For definition questions, the system retrieved response sets from the AQUAINT collection for 28 of the 50 definition questions. Of these 28 questions, 20 contained at least one definition nugget all but one of which contained at least one essential nugget.

3.3 shef12simple

If we analyse the output of the intermediate stages (question typing and information retrieval) then we can

[3] A passage is classed as answer bearing if it comes from a document known to contain the answer and one of the Perl patterns, kindly supplied by Ken Litkowski, matches the text.

Run Tag	*Factoid*	*List*	*Definition*	*Combined*
`shef12madcow`	0.063	0.015	0.171	0.078
`shef12okapi`	0.046	0.033	0.230	0.089
`shef12simple`	0.138	0.029	0.236	0.135

Table 2: Summary results from our three main task entries.

arrive at a maximum attainable score for the final answer extraction stage.

Of the 413 factoid questions the first stage assigned an incorrect type to 53 questions, 27 of these were typed as UNKNOWN so only 26 answers of the wrong type could actually be returned. In total 146 of the questions were assigned the UNKNOWN type, so 267 questions were typed – 241 correctly. The system returned NIL for 191 of the questions so the system was unable to find an answer for 45 of the typed questions. Of the remaining 241 questions 18 have no known answer leaving 223 documents to which the system could return a correct non-NIL answer.

Unfortunately the IR stage was only able to locate answer bearing passages for 131 of the 223 questions correctly processed by the previous stage which means that the maximum obtainable score for the whole system is 0.317 (131/413). The official score for this run is 0.138 (57/413) but this contains fifteen correct NIL responses so we only provided a correct answer for 42 questions giving a score of 0.102 which is 32.2% of the maximum score.

As the answers proposed by this system are named entities, very few are marked as inexact (six are marked as inexact of which only two are missing information) so counting these as correct answers does very little to the overall score increasing it to only 0.153 (63/413).

Of the three runs we submitted this run found the most distinct answers for the list questions – 20 distinct answers compared with 12 for `shef12okapi` and 7 for `shef12madcow`. Unfortuntaely the ability of the system to locate many distinct answers was offset by the fact that for each question many answers are proposed dramatically lowering the precision scores and hence the average F score for the run. For example there are seven known answers to question 2346, *"What countries have won the men's World Cup for soccer?"* for which this run returned 32 answers only two of which were correct giving a recall of 0.286 but a precision of only 0.062. Clearly more work needs to be done to limit the number of answers returned for questions of this type.

Although the score given in Table 2 for the definition section of this run is slightly different from that for the `shef12okapi` run they are actually identical submissions so for more details refer to Section 3.1.

4 Conclusions and Future Work

In comparison to our results from previous years, the performance scores achieved by the three Sheffield systems this year are somewhat disappointing, although it is difficult to have a clear view of their merit without knowing how other systems have also performed this year. Even so, our experiments have thrown up some interesting results, which are suggestive of valuable directions for future work, and we shall close with some comments in this regard.

Firstly, it is striking (perhaps even distressing) to observe that the 'baseline' `shef12simple` system has performed better overall, and on factoid questions in particular, than either `shef12madcow` or `shef12okapi`, the systems which incorporate QA-LaSIE. Looking more closely at the answers produced by the systems, we have found that, in contrast to the short named entity expressions that `shef12simple` typically returns, QA-LaSIE tends to return longer answers, corresponding to the longest amongst the alternative descriptions of the same entity brought together by co-reference resolution. A consequence of this has been that many more of the answers produced by QA-LaSIE have been judged inexact than for `shef12simple`. If we ignore the distinction between answers judged inexact and correct, the performance of the three systems come much more closely into line, and so we expect that the actual performance of the `shef12madcow` or `shef12okapi` systems could be improved by modifying QA-LaSIE's behaviour in terms of its preference for long or short answers. Even given these observations, the simple system has performed sufficiently well that the approach appears worthy of further development/refinement.

Secondly, in regard to the issue of retrieval models, the results obtained using MadCow are better than using Okapi, even given the highly preliminary version of query formulation used. Interestingly, the performance with MadCow is better despite the fact that it returns answer bearing passages for fewer of the questions than Okapi. This result may relate to the fact that MadCow passages are always single sentences, so that the volume of irrelevant text presented to the answer extraction system as 'noise' may be less. The results suggest that boolean retrieval is a worthwhile direction for further research, both in relation to better question analysis and query formulation, and also regarding what is the best size of passage for indexation and retrieval.

Thirdly, the results for the definition system look promising. In future work, we hope to address the following: general improvement of the definition patterns; relaxation of the filters used for the identification of answer-bearing passages; and development of a syntactic-based technique that prunes a parse tree in order to extract definition phases from answer-bearing sentences.

Finally, the work on automatically identifying name expressions within AQUAINT, and gathering these expressions into equivalence classes, has considerably improved our capabilities for processing name expressions. So far, the results of this work have only been used in the query formulation process for boolean retrieval. However, using this approach, we could instead seek to pre-identify all named entities within AQUAINT, and fold the results of this analysis into the inverted index created for boolean retrieval. Then, for example, given a *who* question, we might restrict retrieval to address only those sentences which are known in advance to contain a `person` entity. Likewise, having identified a name such as *President Clinton* in the question, we might use the index to directly access all sentences known to contain an occurrence of any variant of this name, rather than just searching for sentences that contain the words of this name and its variants,

References

Steve Abney, 1997. *The SCOL Manual*. University of Tübingen.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-Intensive Question Answering. In *Proceedings of the Tenth Text REtrieval Conference*.

Mark A. Greenwood, Ian Roberts, and Robert Gaizauskas. 2002. The University of Sheffield TREC 2002 Q&A System. In *Proceedings of the 11th Text REtrieval Conference*.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2000. Question Answering in Webclopedia. In *Proceedings of the 9th Text REtrieval Conference*.

Kevin Humphreys, Robert Gaizauskas, Mark Hepple, and Mark Sanderson. 1999. University of Sheffield TREC-8 Q & A System. In *Proceedings of the 8th Text REtrieval Conference*.

Julian Kupiec. 1993. MURAX: A Robust Linguistic Approach for Question Answering Using an On-Line Encyclopedia. In *Research and Development in Information Retrieval*, pages 181–190.

George A. Miller. 1995. WordNet: A Lexical Database. *Communications of the ACM*, 38(11):39–41, November.

Ian Roberts and Robert Gaizauskas. 2004. Evaluating passage retrieval approaches for question answering. In *Proceedings of 26th European Conference on Information Retrieval*. To Appear.

S. Robertson and S. Walker. 1999. Okapi/Keenbow at TREC-8. In *Proceedings of the 8th Text REtrieval Conference*.

Sam Scott and Robert Gaizauskas. 2000. University of Sheffield TREC-9 Q & A System. In *Proceedings of the 9th Text REtrieval Conference*.