# A Pattern Based Approach to Answering Factoid, List and Definition Questions

**Mark A. Greenwood and Horacio Saggion**
Department of Computer Science
University of Sheffield
Regent Court, Portobello Road
Sheffield S1 4DP UK
{m.greenwood, saggion}@dcs.shef.ac.uk

**Abstract**

Finding textual answers to open-domain questions in large text collections is a difficult problem. In this paper we concentrate on three types of questions: factoid, list, and definition questions and present two systems that make use of regular expression patterns and other devices in order to locate possible answers. While the factoid and list system acquires patterns in an off-line phase using the Web, the definition system uses a library of patterns identified by corpus analysis to acquire knowledge in an on-line stage for each new question. Results are reported over the question sets from the question answering track of the 2002 and 2003 Text REtrieval Conference (TREC).

## 1 Introduction

Finding textual answers to open-domain questions (Hirschman & Gaizauskas, 2001) in large text collections, such as AQUAINT[1], is a challenging problem. Unlike Information Retrieval (IR) which aims at providing documents satisfying users information needs expressed in the form of a query, Question Answering (QA) aims at providing users with short text units that answer specific well formed natural language questions.

There are an infinite number of questions users can ask, in this study, however, we concentrate on those question types for which clearly defined evaluation methods exist, namely those questions used in the question answering evaluation of the TREC (Voorhees, 2002). These questions fall into three categories:

- Factoid questions which require a single fact as answer. These include question such as *"Who is Tom Cruise married to?"* and *"What lays blue eggs?"*.
- List questions which require multiple facts to be returned in answer to a question. Examples are *"Name 22 cities that have a subway system"* and *"List 16 companies that manufacture tractors"*.
- Definition questions, such as *"What is aspirin?"*, which require textual answers that cover essential (e.g., *aspirin is a drug*) as well as non-essential (e.g., *aspirin is a blood thinner*) descriptions of the *definiendum* (i.e., the term to be defined).

No attempt is made by the system detailed in this paper to answer other question types, for example speculative questions, such as *"Is the airline industry in trouble?"* are not handled.

The system described in the body of this paper makes no attempt to analyse the whole AQUAINT collection in the search for an answer, relying instead on the Okapi (Robertson & Walker, 1999) IR engine as an effective filter between AQUAINT and the answer extraction components. Assuming that the IR system is able to find at least one relevant document then the system will continue to process the documents in the hope of finding an answer to the question.

Two different components are used for answering factoid (including list) questions and definition questions. Both components make use of patterns as strategies for locating possible answers. While the factoid system acquires patterns from the Web in an off-line phase, the definition system uses a library of patterns identified by corpus analysis to acquire knowledge about the *definiendum* in an on-line stage

[1]AQUAINT consists of over 1 million texts from the New York Times, the AP newswire, and the English portion of the Xinhua newswire totaling approximately 3.2 gigabytes of data.

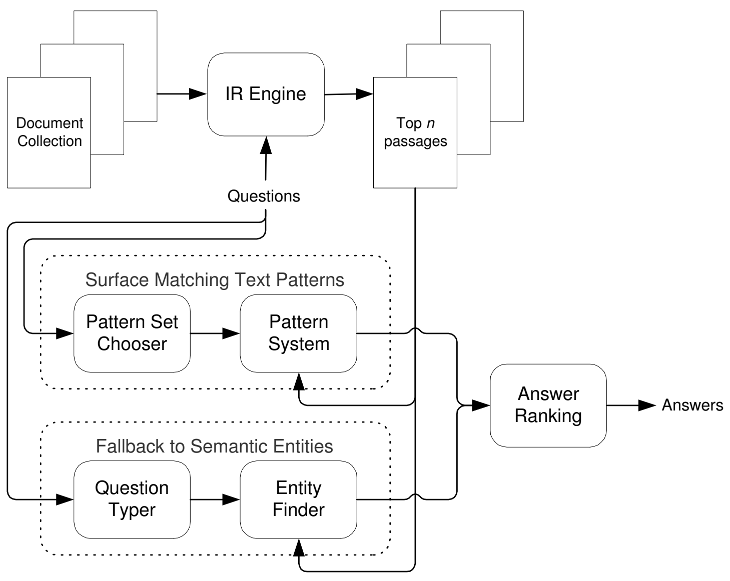

Figure 1: System architecture showing both the pattern and fallback systems.

for each new question. The query used for document retrieval is composed of all the tokens in the question, for definition questions query expansion is also performed.

In the remainder of this paper we detail the approaches for answering each type of question and report evaluation results against both the TREC 2002 and 2003 test sets.

## 2 Answering Factoid and List Questions

The system used for answering factoid and list questions, shown in Figure 1, consists of three main phases:

- The question is used as the query for Okapi to retrieve the top $n$ relevant documents.
- The question and retrieved documents are then used to find possible answers via both the surface matching text patterns and the fallback system.
- Answer ranking then produces a final combined answer list from which as many answers as required can be returned.

This pattern based approach to factoid QA is derived from earlier work presented in (Greenwood & Gaizauskas, 2003). The majority of the system is concerned with the acquisition of answer patterns using a set of question-answer pairs. The patterns which are acquired can then be used to locate answers for new unseen questions. In what follows we describe how the patterns are acquired and how they are used to find possible answers.

### 2.1 Learning Pattern Sets

Learning a pattern set is a two stage process of acquisition and analysis. The easiest way to describe both stages of the process is though an example and for easy comparison with other papers (including (Greenwood & Gaizauskas, 2003), (Ravichandran & Hovy, 2002) and (Soubbotin & Soubbotin, 2001)) we use the example *"When was X born?"*. Given this the acquisition algorithm is as follows:

1. We collect together twenty example questions, of the correct type, and their associated answers.
2. For each example question we produce a pair consisting of the question term and the answer term. For example "Abraham Lincoln"- "1809".
3. For each example the question and answer terms are submitted to Google, as a single query, and the top 10 documents are downloaded[2].

[2]The documents are actually downloaded from Google's cache to guarantee that we use the version of the page indexed by Google.

4. Each retrieved document then has the question term replaced by the single token `AnCHoR`.

5. Depending upon the question type other replacements are then made. In this example all dates, locations, organisations and person names are replaced by representative tags `DatE`, `LocatioN` `OrganizatioN` and `PersoN`. For other question types differing entities may be replaced by tags. If any of these tags replace text which contains the answer term then a compound tag, such as `AnSWeRDatE`.

6. Any remaining instances of the answer term are then replaced by `AnSWeR`.

7. Sentence boundaries are determined and those sentences which contain both `AnCHoR` and `AnSWeR` are retained.

8. A suffix tree (Ukkonen, 1995) is constructed using the retained sentences and all repeated substrings containing both `AnCHoR` and `AnSWeR` and which do not span a sentence boundary are extracted.

This produces a set of patterns, which are specific to the question type. Continuing with the example of *"When was X born?"* a selection of patterns are:

```
from AnCHoR ( AnSWeRDatE - DatE )
     AnCHoR , AnSWeRDatE -
- AnCHoR ( AnSWeRDatE
from AnCHoR ( AnSWeRDatE -
```

Unfortunately these patterns contain no information about how accurate they are when used to answer unseen questions, so the second stage analyses the patterns, removing those which are of little value in answering unseen questions. The analysis algorithm is then as follows:

1. A second set of twenty question-answer pairs are collected and for each question, the question term alone is submitted to Google and the top ten documents are downloaded.

2. Within each document the question term is replaced by `AnCHoR`.

3. The same replacements as carried out in step 5 of the acquisition phase are also performed on the document in this stage and a table is constructed of the inserted tags and the text they replace.

4. Each of the previously generated patterns is converted to a standard regular expression designed to capture the answer text, giving expressions such as:

   ```
   from AnCHoR \( (DatE) - DatE \)
   AnCHoR , (DatE) -
   - AnCHoR \( (DatE)
   from AnCHoR \( (DatE) -
   :  AnCHoR , (DatE) -
   ```

   These expressions allow us to easily retrieve the single token which `AnSWeR` (or one of it's extended forms such as `AnSWeRDatE`), in the original pattern would have matched against.

5. Each of the previously generated patterns are then matched against each sentence containing the `AnCHoR` tag. Along with each pattern, $P$, two counts are maintained: $C_a^P$, which counts the total number of times this pattern has matched against the text and $C_c^P$, which counts the number of matches which had the correct answer or a tag which expanded to the correct answer as the text extracted by the pattern.

6. After a pattern, $P$, has been matched against the sentences if $C_c^P$ is less than five it is discarded otherwise the precision of the pattern is calculated as $C_c^P/C_a^P$ and the pattern is retained if its precision is greater than 0.1[3].

Using this method to produce analysed patterns for the question type *"When was X born?"* gives patterns such as those in Table 1, which can now be used to locate answers to unseen questions.

The current system includes pattern sets for the question types: *What is the abbreviation for X?*; *When was X born?*; *What is the capital of X?*; *What country is X the capital of?*; *When did X die?*; *What does X stand for?*

[3]These cut-off values were adopted based on empirical observations made during development.

| *Pattern* | *Precision* |
|---|---|
| `AnCHoR \( (DatE) - DatE \)` | 0.909 |
| `AnCHoR \( (DatE) -` | 0.909 |
| `AnCHoR \( (DatE)` | 0.738 |
| `AnCHoR (DatE) - DatE` | 0.700 |

Table 1: Analysed patterns for questions of the form *"When was X born?"*.

| *Pattern Set* | *% Correct* | *% Wrong* | *% Unanswered* |
|---|---|---|---|
| *What is the abbreviation for X?* | 78 | 7 | 15 |
| *When was X born?* | 60 | 8 | 32 |
| *What is the capital of X?* | 29 | 61 | 10 |
| *What country is X the capital of?* | 55 | 40 | 5 |
| *When did X die?* | 54 | 1 | 45 |
| *What does X stand for?* | 21 | 30 | 49 |

Table 2: Evaluation of the individual pattern sets.

In all of these sets the names of people and organizations as well as dates and locations are replaced by associated tags. The pattern set for questions of the form *"What does X stand for?"* also involves replacing noun chunks with the tag `NounChunK`.

## 2.2 Pattern-Based Answer Extraction

One of the problems with the use of pattern sets for QA is determining which set to use to answer a given question. Our current method of selecting the correct pattern set relies on associating with each set a group of regular expressions which will match against questions suitable for answering with the pattern set and which will also extract the `AnCHoR` term from the question. For example with questions asking when someone was born, regular expressions covering at least the following formulations are available:

- *When was X born?*
- *What year was X born?*
- *What date was X born?*
- *In which year was X born?*
- *X was born in which year?*
- *On what date was X born?*

The sets of regular expressions are applied to the question, in no particular order, until a match is found or until all the sets have been exhausted. If the question matches against a regular expression then the `AnCHoR` is extracted and is passed, along with the associated pattern set, to the module responsible for locating possible answers.

Using a pattern set to find answers to a question is extremely simple. Each document retrieved by Okapi has the question term replaced by `AnCHoR` and other associated replacements are made (the same as in step 5 of the acquisition phase).

Each pattern within the set is then matched against the sentences containing `AnCHoR` and for each successful match the token captured by the expression (or if the token is a tag then the text the tag replaced) is stored along with the precision of the pattern. When all the patterns have been applied to all the sentences any answers located are passed to the answer ranking module which returns the required number of answers to the user.

## 2.3 Pattern Set Accuracy

Each analysed pattern set was evaluated over one hundred unseen examples with relevant documents being downloaded from the Internet via Google, results can be seen in Table 2. It should be clear from these results that although most of the pattern sets can correctly answer over 50% of the questions a couple of the sets struggle, answering less than 30% of the questions.

We believe this problem is related to the analysis stage in which only the question term is submitted to Google, this means that unless the answer term is highly related to the question term few of the retrieved documents contain the answer and hence few patterns will be retained. For example with questions of the type *"What is the capital of X?"* it is likely that there will be many documents containing the name of the country that do not contain the name of the capital. A possible solution to this problem is to use the entire question when analysing the patterns rather than just the question term, in this instance *"capital"* would also be a search term increasing the chance of finding appropriate documents.

## 3 Fallback to Semantic Entities

As developing pattern sets is a time consuming task it has not yet been possible to develop enough pattern sets to cover the variety of questions that may be asked. A simple fallback system is therefore required to fill this hole until further pattern sets are developed. The approach taken was to assume that the answer to every question will be one or more entities from a fixed set of semantic types. The entity types which can currently be recognised include the standard named entity categories from the MUC evaluations (person, organization, locations, times, dates and monetary amounts) as well as a wider class of measurements (space, mass, duration, etc.) and other types frequently seen in example questions (languages, nationalities, planets, Gods, state flowers, etc.). The following sections provide more details of exactly how the system attempts to answer questions. For a more detailed description of this fallback system interested readers should consult (Greenwood, 2004)

### 3.1 Question Typing

The first stage of this system involves determining the expected answer type of the question. This is achieved through a set of hand coded rules working over the question text. Each rule precondition specifies a number of keywords which the text of the question has to match in order to be classified as one of the specified answer types. As an example, if the keyword *"author"* is found in the question text, then the answer type of the question is *"person"*. Defining these rules is a relatively simple but time consuming task. Even as a fallback system it should be clear that there are many questions which this system will be unable to answer specifically those questions for which the answer is not an entity of a type the system is capable of recognising. If a question is not assigned a type by the rules then a default type of UNKNOWN is assigned.

### 3.2 Locating Possible Answers

Locating all the possible answers within a document simply involves extracting all entities of the expected type, unless they fail one of the following conditions:

- The document which contains the answer entity must also contain all the entities found in the question, and
- the answer entity must not overlap (ignoring stopwords) with the question.

The remaining entities are then grouped together using the equivalence test (Brill et al., 2001):

*two answers are said to be equivalent if all of the non-stopwords in one answer are present in the other answer or vice versa.*

When two answers are deemed to be equal then the longest realisation of the answer is the one retained. The resulting answer list is then passed to the answer ranking module to determine how many, if any, of the answers will be returned to the user.

## 4 Answer Ranking

If both approaches have located possible answers then there will be two competing answer lists. As the pattern based approach is more restricted in the answers it can propose these are always preferred over those found by the fallback system, so the answers from the fallback system are discarded[4]. Depending on the source of the final answer list it is sorted in one of the following ways:

- if the answer list was generated by the pattern based approach then the answers are sorted based on the highest precision of the patterns which located the answers and then (if necessary) the highest rank of the documents in which the answer was found.

[4]Implementations of this system clearly do not need to produce an answer list using the fallback system if the pattern based approach has already yielded one or more possible answers.

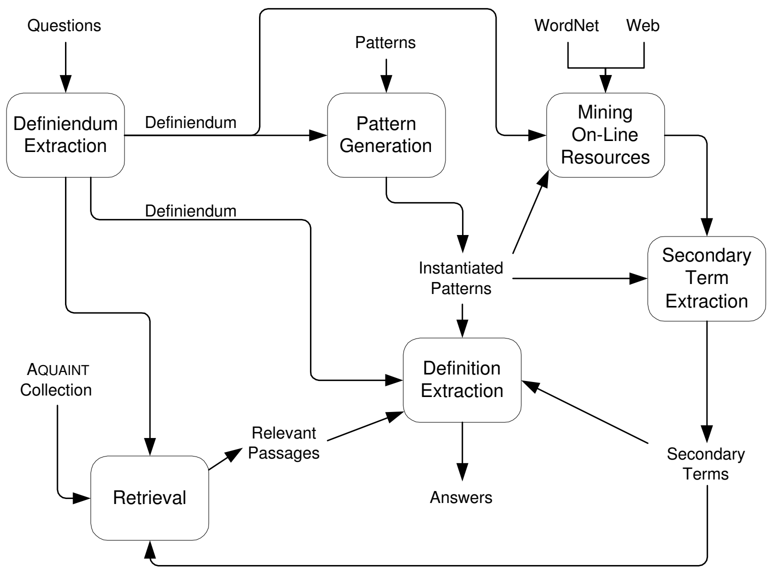

Figure 2: Definition system components

- if the answer list was generated by the fallback system then the answers are sorted based on the number of occurrences of the answer that were found and then (if necessary) the highest rank of the documents in which the answer was found

If the current question is a simple factoid question then the first answer from the sorted list is returned if it is a list question then the entire answer list is returned. If we were unable to generate an answer list (i.e. both the pattern approach and the fallback system were unable to locate any answers) then the system returns NIL.

## 5 Answering Definition Questions

Answering definition questions requires a different strategy to that used for answering factoid and list questions. This is because definition questions provide very little information which can be used to help find relevant definition-bearing passages in the text collection apart from the *definiendum*.

Our method for identifying answers to definition questions relies on using secondary terms to restrict the search for definition-bearing passages, and a limited number of *"seed"* patterns to extract definitions. Secondary terms, mined from on-line sources, are words which co-occur with the *definiendum* in definition contexts outside the target text collection The terms can therefore be used to refine the search for definition-bearing passages in the target collection. For example, a term like "aspirin" occurs in 1108 sentences in the AQUAINT collection most of which are not definition-bearing sentences. If, however, we know from external sources that the term "analgesic" usually occurs with "aspirin" in a definition context, then both "aspirin" and "analgesic" can be used as search terms. This results in finding only 8 sentences in the AQUAINT collection.

Having reduced the search space to a small set of passages, definition patterns and filters, involving the secondary terms, are used to select the best non-redundant answers. This process is shown in Figure 2.

## 6 Acquisition of *Definiendum* Related Knowledge from On-Line Sources

Obvious sources for mining definition related knowledge are dictionaries and encyclopedias. These resources are particularly appropriate when trying to create a library of general definition patterns. In our case, however, we aim to obtain knowledge about particular terms that may or may not appear in these resources. In such cases, the redundancy of the web (Brill et al., 2001) can also be exploited. We rely on the WordNet lexical database (Miller, 1995), the web site of the Encyclopedia Britannica[5], and the Web[6] for finding definition-bearing passages for the term sought. The process consists of the

[5] `http://www.britannica.com`

[6] The Google Web API (`http://www.google.com/apis`) is used to download documents from the web.

| Pattern | | Passage | |
|---|---|---|---|
| **Uninstantiated** | **Instantiated** | **Relevant** | **Not Relevant** |
| TERM is a | aspirin is a | *Aspirin is a* weak monotripic acid | *Aspirin is a* great choice for active people |
| such as TERM | such as aspirin | blood-thinners *such as aspirin ...* | Look for travel size items *such as aspirin* and... |
| like TERM | like aspirin | non-steroidal anti-inflammatory drugs *like aspirin ...* | A clown is *like aspirin*, only he works twice as fast |

Table 3: The first column contains uninstantiated definition patterns. The second column contains the pattern instantiated with the term "aspirin". The third column contains definition-bearing passages matching the pattern. The last column contains passages which match the pattern but are not definition bearing.

following steps:

1. *Definiendum* extraction: the term, for which a definition is sought, is identified in the question;
2. Pattern generation: a number of definition patterns containing the sought term are generated;
3. Document Retrieval: texts containing the patterns are retrieved from the web and WordNet;
4. Term extraction: words that co-occur with the sought term, in sentences matching one or more of the patterns, are extracted.

Each of these steps is described in detail in the following subsections. All linguistic analysis is performed using GATE (Cunningham et al., 2002) and components from the LaSIE (Humphreys et al., 1998) system.

### 6.1 *Definiendum* Extraction

Since the *definiendum* can be a complex linguistic unit such as a term or named entity (e.g., "What is Aum Shirikyo?"), we rely on named entity recognition, part-of-speech tagging, and parsing to extract it from the question. Named entity recognition and classification is based on gazetteer list lookups and regular expression matching. The result of this analysis together with part-of-speech information is used as input to a chart parser which uses a context-free grammar specially adapted to parse questions. The result is a syntactic representation of the question (which may be a set of unconnected chunks) from which the *definiendum* (the rightmost noun phrase in the parsing tree) can be extracted.

### 6.2 Pattern Generation

Following (Saggion & Lapalme, 2002), a repository of 50 *"seed"* definition patterns have been manually assembled through an analysis of definition bearing texts. The patterns used in this work are rather superficial, containing only lexical information. The idea of using reliable seed patterns has been exploited by (Hearst, 1992) to identify lexical semantic relations such as hyperonymy, in order to construct ontologies from corpora.

The patterns are used by instantiating them for a given *definiendum* as a precursor to further linguistic processing. Table 3 shows a selection of definition patterns, their instantiation after the *definiendum* is known, and passages containing an exact pattern match.

### 6.3 Mining On-Line Sources

A collection of texts, from which secondary terms will be extracted, is built using a number of electronic resources:

- WordNet: all glosses associated with meanings of the *definiendum* are extracted along with any hypernyms (super-ordinate word) associated with the *definiendum*.
- Encyclopedia Britannica: all pages containing the *definiendum* are downloaded from the encyclopedia web site (because we do not subscribe to this resource, we have only access to incomplete information from this site).

| ***Definiendum*** | **WordNet** | **Britannica** | **Web** |
|---|---|---|---|
| Aum Shirikyo | | | group; groups; cult; religious; japanese; stages; terrorist; early; doomsday; etc. |
| aspirin | analgesic; anti-inflammatory; antipyretic; drug, etc. | inhibit; prostaglandin; ketoprofen; synthesis; etc. | drugs; drug; blood; ibuprofen; medication; pain; etc. |

Table 4: The first column lists the *definiendum*. The second column shows terms associated with the *definiendum* which were found in WordNet. The third column contains terms associated with the *definiendum* found on the Britannica web site. The last column contains the terms found on the Web.

- Web: for each instantiated pattern, English language HTML pages containing an instantiated pattern (e.g., "aspirin is a") are downloaded. We rely on Googles exact match facility to ensure the entire pattern is present in the page. Note that multiple patterns can download copies of the same page although only one copy of each page is retained for further processing.

### 6.4 Secondary Term Extraction

Secondary terms are obtained from the texts found in the previous step by performing the following processes: tokenisation; sentence splitting; part-of-speech tagging; and pattern matching of the instantiated definition patterns. Three different methods are used depending on the source of the text:

- For each sentence in a WordNet gloss all nouns, verbs, and adjectives are extracted and stored along with any hypernyms of the *definiendum*.
- For each sentence in the document downloaded from the Encyclopedia Britannica the nouns, verbs, and adjectives are extracted only if the sentence contains the *definiendum*. Each extracted term is stored along with its frequency of occurrence.
- For each sentence in the other Web documents: if the sentence contains a definition pattern, then the nouns, verbs, and adjectives are extracted and stored along with their frequency of occurrence.

Terms extracted from the Encyclopedia Britannica and the Web are sorted in decreasing order of their frequency of occurrence. Table 4 shows a selection of secondary terms extracted from the different sources for the terms "aspirin" and "Aum Shirikyo" (a Japanese sect).

A list of secondary terms to be used for query expansion (with a maximum of $n$ elements) is created in the following way: firstly all terms extracted from WordNet ($m$ terms) are included in the list; a maximum of $(n - m)/2$ terms from the Encyclopedia Britannica with a frequency greater than one are added to the list; and finally terms extracted from the Web with a frequency greater than one are appended to the list until the maximum of $n$ is reached.

The order of terms in the list reflects, to a certain extent, the degree of confidence in the sources from which they were extracted. Also the more a term is used in a particular "definition" context the more we believe the term is associated with the *definiendum* hence we prefer terms extracted from definitions of the *definiendum*, WordNet, followed by those extracted from a definition of any term, Encyclopedia Britannica, and finally those that simply co-occur with the *definiendum* regardless of context, the Web.

## 7 Locating Possible Definitions

We construct a query consisting of the tokens contained in the question merged with the list of secondary terms found during the knowledge acquisition step. This query is then submitted to Okapi and the 20 most relevant passages from the AQUAINT collection are retrieved.

Linguistic analysis of each passage is then performed, consisting of: tokenisation, sentence splitting, dictionary look-up for the *definiendum* and any of the secondary terms, and pattern matching using the instantiated patterns obtained during knowledge acquisition. We restrict our analysis of definitions to the sentence level. A sentence is considered definition-bearing if it matches a definition pattern or if it contains the *definiendum* and at least three secondary terms (this heuristic was determined after several experiments on training data). In an effort to avoid the inclusion of unnecessary information, which may be present in a definition-bearing sentence, we discard the sentence prefix which does not contain either the *definiendum* or any of the secondary terms.

One of the problems faced when extracting definitions from huge text collections is that of redundancy: in collections of newswire texts the same pieces of information may appear many times. We address this problem by measuring the similarity of a candidate definition to the previously extracted definitions. A vector representation of the extracted candidate definition, consisting of its terms and term frequencies, is created. If the current answer candidate is too similar to any of the previous extracted answers, it is ignored. The similarity score is the cosine between the two vector representations, and two vectors $v_1$ and $v_2$ are considered similar if $cosine(v_1, v_2) > threshold$ (the $threshold$ was set after a number of experiments on a training set). Semantic similarity is not considered in this approach.

## 8 Evaluation

In the following sections we describe the results obtained by our system over both the TREC 2002 and 2003 QA test sets.

### 8.1 Factoid and List Questions

Before entering the system in the TREC 2003 QA evaluation we evaluated the system over the questions used in the 2002 evaluation. The system correctly answered 26% (130/500) of these questions. Only 33 questions were suitable to be answered using the pattern sets and in all but one case the question term, AnCHoR, was correctly identified. Relevant documents within AQUAINT were located for 20 of these questions. Unfortunately the pattern sets only located answers for 7 of these 20 questions of which only 2 answers were correct.

The TREC 2003 QA factoid and list subtasks consisted of finding answers for 413 factoid and 37 list questions. The evaluation metric for factoid questions was answer-accuracy (the percentage of correct answers found by the system) and for list questions was the mean F-score over all list questions (where precision and recall are equally weighted).

Of the 413 factoid questions 12 questions were suitable for answering with the pattern sets, unfortunately none of the patterns selected any answers, correct or otherwise. Using the fallback system 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 questions to which the system could return a correct non-NIL answer. Unfortunately Okapi was only able to locate answer bearing passages for 131 of the 223 questions correctly assigned a type, which means that the maximum obtainable score for the whole system is 0.317 (131/413). The official TREC score was 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. The average accuracy over all the systems evaluated (54 runs) was 0.700 (best), 0.177 (median), 0.034 (worst).

Similar problems occurred when the system attempted to answer the list questions. Over the 37 questions the system returned 20 distinct correct answers achieving an F-score of 0.029 (compared to the average F-score, obtained by the systems evaluated, of 0.396 (best), 0.069 (median), and 0.000 (worst)). Unfortunately the ability of the system to locate a reasonable number of distinct answers was offset by the fact that for each question many answers (i.e. all those found) are proposed dramatically lowering the precision and hence the F-score. For example there are seven known answers, within AQUAINT, to the question *"What countries have won the men's World Cup for soccer?"* for which the system returned 32 answers only two of which were correct. This gave a recall of 0.286 but a precision of only 0.062.

The results from both evaluations show that the system still requires improvement. Currently the acquired patterns sets contributing little, if anything, to the performance of the system with most of the questions which were answered correctly answered by the fallback system.

$$NR = \frac{\text{number of essential nuggets returned}}{\text{number of essential nuggets}}$$

$$\text{allowance} = 100 * \text{number of essential or non-essential nuggets returned}$$

$$\text{length} = \text{number of non-white-space characters in the answer set}$$

$$NP = 1\text{: if length} < \text{allowance}$$

$$NP = 1 - \frac{\text{length-allowance}}{\text{length}}\text{: if length} \geq \text{allowance}$$

$$F = \frac{26 * NP * NR}{25 * NP + NR}$$

Figure 3: Formulas used to score answers to definition questions.

### 8.2 Definition Questions

The TREC QA 2003 definition subtask consisted of finding answers for 50 definition questions. The set consisted of 30 “Who” questions (e.g., *“Who is Aaron Copland?”*) and 20 “What” questions (e.g., *“What is a golden parachute?”*). The TREC assessors created, for each question, a list of acceptable information nuggets (pieces of text) from all submitted answers and information discovered during question development. Some nuggets are considered essential (i.e., a piece of information that should be part of the definition) while others are considered non-essential. During evaluation, the assessor takes each system response and marks all essential and non-essential nuggets contained in it. A score for each question consists of nugget-recall (NR) and nugget-precision (NP), see Figure 3. These scores are combined using the F-score measure with recall five times as important as precision. We obtained an F-score of 0.236. The average F-score over all systems (54 runs) is 0.555 (best), 0.192 (median), 0.000 (worst); placing our definition system above the median.

In 5 cases the *definiendum* extraction component extracted an incorrect term, for example for the question “*Who was Abraham in the Old Testament*” the system extracted the *definiendum* “*Abraham in the Old Testament*” instead of “*Abraham*”. This seriously affects the performance of the extraction system because of the particular way in which patterns are created with the *definiendum*.

Our system provided answer sets for 28 of the 50 questions (56%). For these 28 questions, 23 (82%) contained at least one definition nugget and all contained at least one essential nugget. The system answered 12 “Who” questions (40% of this type) and 11 “What” questions (55% of this type).

Considering the knowledge acquisition step, WordNet provided relevant secondary terms for only 4 questions (8%). The Britannica web site helped with relevant secondary terms in only 5 questions (10%). The Web, however, helped to find relevant secondary terms for 39 questions (78%). In cases where the knowledge acquisition system failed in finding relevant terms or found only a few irrelevant terms, the answer extraction system also failed. This is due to the filters that are used to determine if a sentence is extracted. The patterns as they are implemented in the current version of the system allow for little variation, this has an impact on both the knowledge acquisition and extraction steps.

In the light of nuggets identified by NIST analysts, a more ‘lenient’ and optimistic analysis can be done of the answers provided by (and omissions of) our system. Our system returned no answers in 22 cases. After close examination of the documents returned by the Okapi retrieval system, we can conclude that in 15 cases (68%) the documents contain some of the nuggets required to answer the question. The system failed to return answers because the definition patterns and filters were far too restrictive to cover these definitions (recall problem), a problem that we are presently addressing.

## 9 Related work

The concept of “answer patterns” is not new in the field of QA. Mostly these patterns have constituted only a small part of a system, such as those used in (Harabagiu et al., 2000) which were used solely for answering definition questions. The first attempt at using patterns as the main approach to open-domain QA can be traced back to a system (Soubbotin & Soubbotin, 2001) entered in the 2001 TREC QA evaluation. Of the 289 correct answers returned by this system 193 (66%) were located using acquired answer patterns, unfortunately details of exactly how the other 96 questions were answered is scant.

Such was the interest in this pattern based approach that at least one group (Ravichandran & Hovy, 2002) choose to implement their own version. This implementation achieved similar results to the original system but more details were made available allowing further work to be carried out by other interested parties. The unfortunate point of this implementation was that no attempt was made to build an entire QA system, rather the intent was to show that the approach worked well over the question types evaluated.

The approach to learning answer patterns used in this paper is an extension of that given in (Greenwood & Gaizauskas, 2003) which attempts to find more generalised patterns than those in either of the two previous systems via the use of more detailed natural language analysis, i.e. named entity recognition and noun chunking. This approach is very similar to (Ravichandran & Hovy, 2002) achieving similar results over specific question types but allowing patterns to be acquired for a large range of question types.

Related to our approach for answering definitions is the DefScriber system (Blair-Goldensohn et al., 2003) that combines both world knowledge in the form of definitional predicates (genus, species, and non-specific) and data-driven statistical techniques. World knowledge about the predicates is created via machine learning techniques over annotated data. Data-driven techniques including vector space model and cosine distance are used to exploit the redundancy of information about the "*definiendum*" in non-definitional Web texts. We differ from this approach in two aspects. Firstly we do not rely on annotated data to acquire knowledge and secondly we mine information in definition-bearing passages to ensure that only terms relevant for definitions are found. Nevertheless, we employ similar statistical techniques to identify similarities across sentences.

Also related to our research is the work by (Fleischman et al., 2003) who create pairs of concept-instances such as "*Bill Clinton-president*" mining newspaper articles and web documents. These pairs constitute pre-available knowledge that can be used to answer "*Who is ...?*" questions. They use lexico-syntactic patterns learnt from annotated data to identify such pairs. We differ in that we learn knowledge in a targeted on-line way, without relying on annotated data. The knowledge acquired can be used to answer two different types of definition questions "Who" and "What".

## 10 Conclusion and Future Work

In this paper we have presented a pattern-based method for answering factoid, list, and definition questions from large text collections that relies on knowledge mined from on-line sources. The pattern-based method for answering factoid and list questions is combined with a simple method which assumes that all questions require a named entity as answer, the most frequent entity (of the correct type) in the retrieved documents being proposed as the answer. The definition extraction method combines definition patterns with the presence of secondary terms likely to appear in definition-bearing passages.

When answering factoid questions we noted that as the patterns have been acquired from the Web, they do not necessarily reflect the writing style of documents of in a collection such as AQUAINT. When answering definition questions we have identified that the filters used to determine if a passage is definition-bearing, are far too restrictive and so they should be relaxed.

Results obtained on the recent TREC QA 2003 evaluation indicate many avenues of future research. For the factoid and list QA system these could include:

- acquiring a wider range of pattern sets to cover more question types;
- experiments to gauge the affect of altering the document retrieval of the analysis stage to use the full question not just the question term;
- development of a method which given a question and answer (gold standards) attempts to find supporting documents within a closed collection from which patterns are learn;

For the definition QA system future research could include:

- extracted secondary terms for definition questions could be ranked, perhaps using their IDF values, in order to help to eliminate inappropriate definition pattern matches (*aspirin is a great choice for active people*).
- in order to extract better definition strings, a syntactic-based technique that prunes a parse tree could be implemented;
- in order to cross the sentence barrier, coreference information could be used in the extraction patterns;
- in order to improve recall, a relaxation of the definition criteria currently implemented or development of a back-off strategy could be considered.

## References

Blair-Goldensohn, S., McKeown, K. R., & Schlaikjer, A. H. (2003). DefScriber: A Hybrid Approach for Answering Definitional Questions (Demo). In *Processdings of the 26th ACM SIGIR Conference*, Toronto, Canada, July. ACM.

Brill, E., Lin, J., Banko, M., Dumais, S., & Ng, A. (2001). Data-Intensive Question Answering. In *Proceedings of the Tenth Text REtrieval Conference*.

Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*.

Fleischman, M., Hovy, E., & Echihabi, A. (2003). Offline Strategies for Online Question Answering: Answering Questions Before They Are Asked. In *Proceedings of the ACL 2003*, (pp. 1–7). ACL.

Greenwood, M. A. & Gaizauskas, R. (2003). Using a Named Entity Tagger to Generalise Surface Matching Text Patterns for Question Answering. In *Proceedings of the Workshop on Natural Language Processing for Question Answering (EACL03)*, (pp. 29–34), Budapest, Hungary, April 14.

Greenwood, M. A. (2004). AnswerFinder: Question Answering from your Desktop. In *Proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics (CLUK '04)*, (pp. 75–80), Birmingham, UK, January 7.

Harabagiu, S., Moldovan, D., Paşca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Gîrju, R., Rus, V., & Morărescu, P. (2000). FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of the 9th Text REtrieval Conference*.

Hearst, M. A. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of COLING'92*, Nantes.

Hirschman, L. & Gaizauskas, R. (2001). Natural Language Question Answering: The View From Here. *Natural Language Engineering*, 7(4).

Humphreys, K., Gaizauskas, R., Azzam, S., Huyck, C., Mitchell, B., Cunningham, H., & Wilks, Y. (1998). Description of the LaSIE-II system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.

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

Ravichandran, D. & Hovy, E. (2002). Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, (pp. 41–47), Pennsylvania.

Robertson, S. E. & Walker, S. (1999). Okapi/Keenbow at TREC-8. In *Proceedings of the 8th Text REtrieval Conference*.

Saggion, H. & Lapalme, G. (2002). Generating Indicative-Informative Summaries with SumUM. *Computational Linguistics*, 28(4):497–526.

Soubbotin, M. M. & Soubbotin, S. M. (2001). Patterns of Potential Answer Expressions as Clues to the Right Answers. In *Proceedings of the 10th Text REtrieval Conference*.

Ukkonen, E. (1995). On-line Construction of Suffix Trees. *Algorithmica*, 14(3):249–260.

Voorhees, E. M. (2002). Overview of the TREC 2002 Question Answering Track. In *Proceedings of the 11th Text REtrieval Conference*.