Chapter 4: Implementation of the System

4.1 Outline

This chapter first introduces the DLT question answering system that served as the starting point for the software used in this project. The second section describes the pre-processing of the documents in SOK-i. The last section explains how we modified various components of the system for the purpose of this project.

4.2 The DLT Question Answering System

4.2.1 Background

The Documents and Linguistic Technology group at the University of Limerick participated for the first time in the TREC evaluation in 2002 (Sutcliffe, 2003). The DLT system which was entered in TREC-11 (2002) was developed within three weeks and thus was basic. The system managed to answer 9% of questions correctly (excluding inexact and unsupported answers). During 2003, the system was adapted for cross-lingual (French to English) question answering purposes (Sutcliffe, Gabbay and O'Gorman, 2003b), and some of these adaptations were incorporated in the monolingual system when used for TREC-12 (Sutcliffe et al., 2003a). In TREC-12, the system achieved the same level of performance as in TREC-11 (see Table 4.1).

	Run 1	**Run 2**
Correct (factoid)	37 (9%)	33 (8%)
Inexact (factoid)	9 (2%)	3 (0.7%)
Unsupported (factoid)	0	1 (0.3%)
Wrong (factoid)	367 (89%)	376 (91%)
Factoid accuracy	0.09	0.08
Average F score for list questions	0.032	0.034
Average F score for definition questions	0.126	0.133
Final combined score	0.084	0.082

Table 4.1 Performance statistics of the DLT system in TREC-12. Run 1 used the highest_scoring strategy for answer selection. Run 2 used the highest_google scoring.

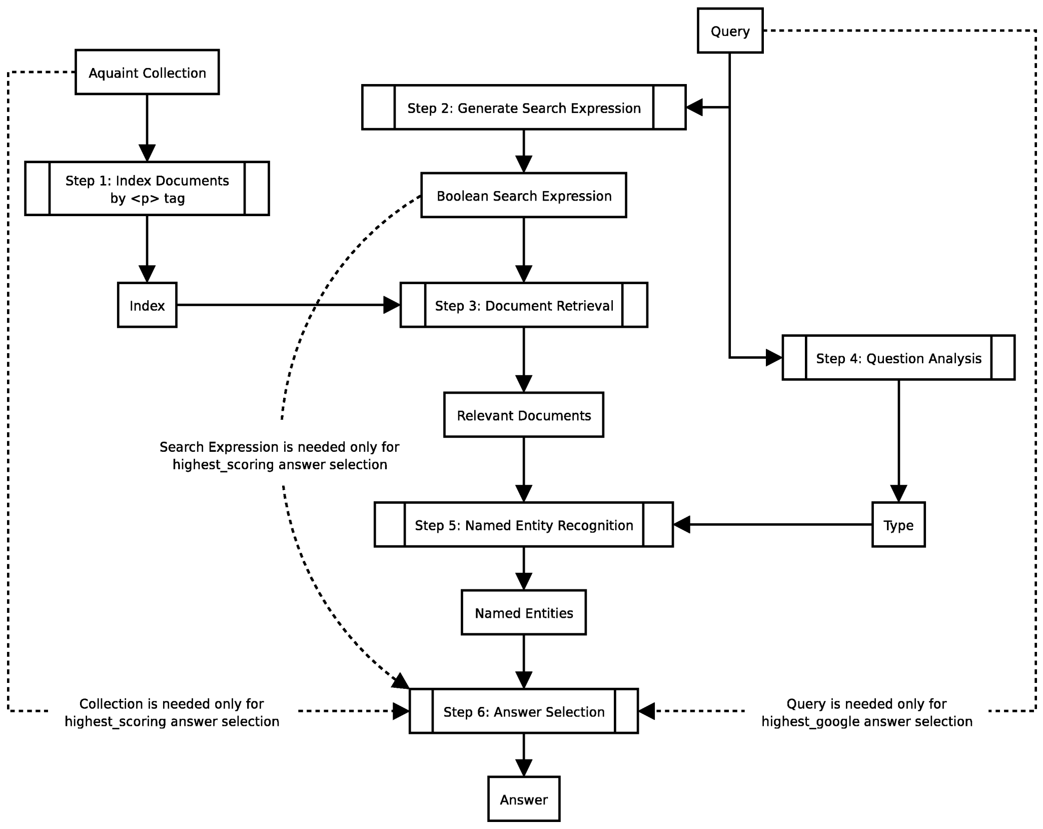
Query
Aquaint Collection
Step 2: Generate Search Expression
Step 1: Index Documents by <p> tag
Boolean Search Expression
Index
Step 3: Document Retrieval
Step 4: Question Analysis
Relevant Documents
Search Expression is needed only for highest_scoring answer selection
Step 5: Named Entity Recognition
Type
Named Entities
Collection is needed only for highest_scoring answer selection
Step 6: Answer Selection
Query is needed only for highest_google answer selection
Answer

Figure 4.1: General architecture of the DLT question answering system used in TREC-12 (White, 2003).

4.2.2 Overall Strategy of the System

The architecture of the DLT system (see Figure 4.1) is common to many other question answering systems in TREC and consists of

- **Question analysis**—determining the type of question and formulating a search query;
- **Document retrieval**—submitting the search query to a search engine indexed on the AQUAINT corpus and returning a list of documents likely to contain answers to the question.
- **Named entity recognition**—searching for named entities appropriate to the question type as identified during question analysis.
- **Answer selection**—deciding which named entity is the right answer.

4.2.3 Processing a Question by the DLT System

The search engine in DLT is dtSearch (dtSearch, 2000) and this was used to index each paragraph (determined by `<p>` tags) in the collection as a separate 'document'. Search is case- insensitive and stopwords were not indexed.

4.2.4 Processing a Question by the DLT System

To illustrate by example, the factoid question in number 1900 in TREC-12, 'What country is Aswan High Dam located in?' (here forth we will use the notation T12-1900), is first classified as a `what_country` type (one of 43 categories based on the interrogative adverbs and keywords, with no parsing), because it starts with the phrase 'What country'.

To formulate the query the system then analyses the question by

- Removing initial words and phrases,
- Recognising capitalised word sequences and expressions within quotation marks,
- Computing alternatives for capitalised word sequences and for acronyms or abbreviations (with or without dots between characters),
- Computing alternative formulations for numbers,
- Removing stopwords,
- Changing the tense of any remaining verbs to simple past.

The remaining terms are assigned an importance score (quotations score 10, capitalised word sequences 9, numbers, pure nouns and verbs 7, superlative adjectives 6, pure adjectives 2, pure adverbs 1 and every other term 5). Terms are then ordered by increasing score with AND operators between them to form a Boolean search.

In the above example, 'What country' are removed as an initial phrase, while 'is' and 'in' are stopwords. 'Aswan High Dam' is a sequence of capitalised words and therefore assigned the score 9. 'located' is a pure verb (cannot have any other part of speech), and assigned the score 7. The resulting search query is thus '`located AND Aswan High Dam`'. This was submitted to dtSearch.

In TREC-12, the number of documents retrieved was set to 30. If no document is returned, the first term in the search query (i.e., the one with the lowest score computed as above) is removed, and the search is repeated. The process is continues until at least one document is found or no terms remain.

To prepare for answer selection, the system marks up query terms in retrieved documents. In the example, the terms are ‘Aswan High Dam’ or ‘aswan high dam’ (the alternatives to all-capitalised words are computed during question analysis) and ‘located’. The system then searches for named entities appropriate for the type of question appearing in documents (paragraphs) containing at least one query term (‘Aswan High Dam’ or ‘aswan high dam’ or ‘located’). In this example, the named entity is a country. The system identifies in the document every country name which appears in a database of countries. When the type of question is unknown (unclassified), the system searches for a general name (a sequence of up to five capitalised words with optional lower case prepositions).

In 2002 the two strategies for answer selection were highest_scoring and most_frequent. In the first strategy the system selects as the answer the named entity that appears in the document with the highest number of unique query terms. In the second strategy the system selects as the answer the named entity that is most frequent across all documents retrieved.

In 2003, a new strategy for answer selection was introduced. This is similar to an algorithm which was suggested by Magnini et al. (2002) and is based on the idea of using the Web search engine Google to validate answers. The system

1. submits the answer candidate to Google with the search query terms, and records the number of hits,
2. submits the answer candidate alone to Google and records the number of hits,
3. divides the value recorded in Step 1 by the value recorded in Step 2.

For factoid questions such as question T12-1900, the candidate with the highest Google score was selected (for list questions the threshold is set to 0.03).

Two of the answer candidates found for T12-1900 were Egypt and Sudan. When 'Egypt' (the correct answer) is submitted to Google with the query terms (`Aswan High Dam AND located`), the hit count returned is 2,960. When 'Egypt' is submitted alone, the hit count is 9,400,000. Thus the ratio of the two values is 0.000314. When 'Sudan' is submitted with the query terms the hit count is 942. When 'Sudan' is submitted alone, the hit count is 3,090,000, and the ratio of the two values is 0.000305. Therefore, 'Egypt' should be selected as the answer.

4.2.5 Answering Definition Questions with the DLT System

For the purpose of this thesis, we had created an initial set of simple lexical patterns to find answers to 'What is X' questions (Table 4.2). The patterns were based on the work available in mid 2003 (Liu, Wee and Ng, 2003; Joho, 1999; Joho and Sanderson, 2000; Hearst, 1992; Klavans and Muresan, 2001), before we learnt about the strategies applied by all the other TREC participants tackling the definition question sub-task for the first time.

TERM is the term for DEF
TERM is the term used to describe DEF
TERM is used to describe DEF
TERM is/was/are/were defined as DEF
TERM, which is/are/was/were DEF
defines TERM as DEF
TERM defines DEF
TERM and other DEF
TERM or other DEF
TERM, a DEF
TERM, the DEF
TERM means DEF
TERM, or DEF

Table 4.2 The set of patterns used by the DLT system in TREC-12 to identify definitions. TERM is the query term. DEF (definition) simply matches the rest of the sentence.

The matching of additional patterns in which the definition is expected to precede the term (for instance, **`DEF is termed TERM`** or **`DEF is known as TERM`**) was not implemented in the DLT system.

The text returned as the answer to a definition question was extracted up to the end of the sentence, although overshooting the exact definition ran the risk of being penalised for exceeding the length allowance. The sentence boundary was defined as a full stop, semicolon or colon. To avoid mistaking abbreviation stops for the end of a sentence (e.g., the stop after ‘J’ in ‘Homer J. Simpson’), the word preceding the stop must be longer than one character if it begins with an upper case letter, and the full stop must be followed by a word beginning with an upper case letter.

Despite the crudeness of this technique and the small number of patterns used, the system managed to retrieve at least one Vital nugget in 15 responses, and at least one Okay nugget in four additional responses (see Tables 4.3 and 4.4 for examples). In at least seven cases no documents were retrieved. In other instances, the question target was misidentified. For example, the system searched for the definition of ‘Ph’ (this of course should be pH) and ‘biology’ in response to the question ‘What is Ph in biology?’.

Question	**Vital nugget**
T12-1957 What are fractals?	sets of complex geometric shapes that look the same over a wide range of scales
T12-2224 Who is Andrew Carnegie?	steel tycoon whose money built more that 1,6000 public libraries in the United States in the early part of the century, is a name that crops up frequently alongside Gates’ these days
T12-2267 Who is Alexander Pope?	the greatest poet of his age, but as a Catholic was unacceptable as laureate to the Protestant House of Hanover

Table 4.3: Example of nuggets retrieved by the DLT system in response to definition questions and evaluated as Vital in TREC-12.

Question	**Okay nugget**
T12-2321 What is Restorative Justice?	mediator establishes a relationship between the offender and the victim
T12-2060 Who is Albert Ghiorso?	leading nuclear experimentalist at Lawrence Berkeley National Laboratory in Calif
T12-2203 What is a quasar?	An object that has died billions of years ago

Table 4.4: Example of nuggets retrieved by the DLT system in response to definition questions and evaluated as Okay in TREC-12.

No attempt was made to classify definition questions as Who or What questions or into Person, Organisation, and Thing categories. Of the 15 responses in which at least one nugget was vital, 11 responses were to Who questions and 4 to What questions, but this may simply reflect the preponderance of Who questions in the test set. Verbal phrases, missing in the set, could have benefited the answers to Who questions, because persons are more likely to be described by their actions. For example, the expected response to T12-2258 'Who is Althea Gibson?' included the following verbal phrases: '`won grand slam titles`', '`broke sports (tennis) color barrier in 1950`', and '`established the Althea Gibson Foundation`'.

The patterns used in DLT did not allow definitions expressed in several parts connected by anaphors such as 'it' or 'this' to be extracted. For example, consider the following definition of 'outbreeding depression': '`One risk of crossing genetically different strains is reduced average fitness (productivity) of F1 or later generation hybrids (Endler, 1977; Wallace, 1981).` **`This phenomenon`** `is called outbreeding depression`'. In TREC and in this project no attempt was made to retrieve such definitions.

4.3 Pre-processing of the Document Collection

Since we focused on simple definitions containing no anaphors, it made sense to split the documents into individual sentences. Marking sentence boundaries also helped in extracting the definition, because when it followed the term (for example, in the pattern **`TERM is defined as DEF`**), the answer we extracted was the text up to the end of the sentence. On the other hand, when the definition preceded the term (for example, in the pattern **`DEF such as TERM`**), the answer was the text from the beginning of the sentence up to the term or up to the words in the pattern (e.g., 'such as' in the last example).

We split the documents, still in HTML format, into sentences, inserting the tag **`SEN`** on a new line to mark a sentence boundary. Appendix B1 shows an example of an HTML document in its source form before splitting took place. Appendix B2 shows the same document after splitting and assigning sentence numbers (see below).

We detected the end of sentences heuristically. A sentence boundary was considered to be

1. A full stop, or exclamation mark, or question mark followed by a space and an upper case letter,
2. A full stop followed by a space and a lower case letter, if the lower case letter is followed by an upper case letter,
3. A full stop followed by a space and an open angled bracket sign (<),
4. A full stop followed by a space and an ampersand,
5. Two newline characters,
6. Two spaces,
7. Header tags (<h1>, <h2>, <h3>, <h4>),
8. Paragraph tags (<p>).

Table 4.5 presents a real example of each of the above rules. After initial splitting of the documents, we observed that the **`SEN`** tag was inserted inappropriately after two-letter abbreviations such as 'e.g.' and author name initials (for example, after the 'R' in 'H.R. Morris'). The tag was also inserted wrongly after the abbreviation 'al.' (in 'et al.'). We corrected this by specifying these cases as exceptions.

The next step was to number the marked sentences in the 1,000 documents. Each sentence was assigned two numbers: the ordinal number of the sentence within the document, and the total number of sentences in the document. As Appendix B2 shows, the ordinal number was inserted in the next line after the **`SEN`** tag, and the total number in the next line after the ordinal number.

All HTML tags were removed from the documents before indexing.

Rule	**Sentence boundary (marked by SEN)**
1. Full stop, space, upper case letter	`'…and another by association with RAMP3 [6, 23].` **`SEN`** `In contrast…'`
2. Full stop, space lower case letter, upper case letter	`'…via CGRP receptors.` **`SEN`** `sCT is a potent agonist at both…'`
3. Full stop, space, <	`'…of the decrease in total energy content.` **`SEN`** `<a href="#bib6">de Boeck et al. (1997)</a> argued that…'`
4. Full stop, space, &	`that during the last 170&nbsp;km.` **`SEN`** `&nbsp;Gill Na<sup>+</sup>,`
5. Two newline characters	`between nutrition, behaviour and environmental stress.` **`SEN`** `<br clear="all">`
6. Two spaces	`Volume 177, Issues 1-4</a>` **`SEN`**
7. Header tags	`<h2> <p>Consumer perceptions of food products involving genetic modification&#x2013;&#x2013;results from a qualitative study in four Nordic countries </h2>` **`SEN`**
8. <p> tags	`'…given the expected continued growth of the industry.` **`SEN`** `<p>Farmed salmon escape from netpens…'`

Table 4.5: examples from SOK-i of sentence boundary detection using eight heuristics.

4.4 Implementation of Changes to the System

4.4.1 Indexing the Document Collection with dtSearch

We used dtSearch version 5.22 for Windows to index the document collection (dtSearch, 2000). dtSearch allows a single file to be indexed on-the-fly as several documents by means of segmentation rules which may be defined by the user. When indexing the documents with dtSearch in CLEF 2003 (Sutcliffe et al., 2003b) and in TREC-12 (Sutcliffe et al., 2003a), we chose the tags **`<DOC>`** and **`<p>`** as the strings indicating the start of a document, respectively. In both cases the tags were in the original documents, unlike the new **`SEN`** tags.

dtSearch includes an editable file with stopwords (or ‘noise words’ in the engine’s terminology). Stopwords are common words, such as ‘is’, ‘the’ and ‘if’, which are normally considered not useful in searches. Excluding stopwords when indexing also reduces the size of the index. However, we chose to delete all the words in dtSearch’s stopword file and index all the words in the documents, because we wanted to be able to search for any exact phrase (e.g. **`TERM is a`**) in experimental configurations of the system.

We availed of these other indexing options in dtSearch: We chose the index to be case insensitive, so as to match any capitalised version of the terms searched. We decided to treat hyphens as spaces (‘cut-off’ would match both ‘cut off’ and ‘cut-off’) and round brackets as searchable characters, because some of the definition patterns included them.

The index totalled 293,847 sentences (“documents”). The number of sentences in a document was 288 on average (median 306) and ranged between 5 and 1,864.

4.4.2 Adaptation of the System

Broadly, the changes made to the original DLT system were either to disable redundant components or to add features to existing modules. All the changes were implemented in Prolog, as was the original system. We used Quintus Prolog 3.4 (Quintus Prolog, 2000). All the experiments were run on a Dell OptiPlex GX1 running Windows 2000 at a clock speed of 500 MHz and having 261Mb RAM.

In the original system, the queries were prepared by reading them into clauses of the form `rdq_query( Number, Query, Type )`. The clauses were then written, twenty at a time, to separate files. We kept this format of input. However, `Type` was now instantiated invariably with ‘definition’ and `Query` was instantiated not with questions but with isolated terms in the exact form in which they were suggested by the salmon researchers or appeared in the FishBase glossary (see Figure 4.2).

```
rdq_query('4073', 'sucking disk', definition ).
rdq_query('4074', 'suction pump', definition ).
rdq_query('4075', 'suctorial', definition ).
rdq_query('4077', 'Sugar cured fish', definition ).
```

Figure 4.2: Example of the format of query input used in the experiments.

When searching the documents, we were interested in an exact phrase match of the query term. Many of the terms consisted of one word, and this obviated the need for query reformulation as described in Section 4.2.3. However, even longer terms should not be reformulated or split, because they then might change their meaning completely (e.g., ‘Length-weight relationship’). Different versions of capitalisation of the term were matched during searching because, as mentioned in the last section, the indexing was case insensitive. We therefore disabled the clauses that parsed and reformulated the query. We also skipped the phase in which the system searched iteratively using increasingly simplified queries until documents were found.

dtSearch was called through a batch script. The script specifies the index that should be used for the search (e.g., AQUAINT or SOK-i documents), the query (with optional operators), the file name into which the results are written, and whether different options should be activated (e.g., stemming). Since we searched only for exact terms, we activated the Boolean search option and de-activated the fuzzy searching and auto-stemming options.

We discovered that if we searched for a term, our version of dtSearch did not match the term when it was enclosed in round brackets. For example, searching for ‘redd’ would not match ‘(redd)’. This was the result of indexing brackets as searchable characters. To overcome the problem, we added the bracketed form to the query line in the batch script. For example, when searching for the term ‘artificial photoperiod’, the query in the batch script read ‘`artificial photoperiod OR (artificial photoperiod)`’.

```
C:\resources\Salmon_numbered_sen_new\c00afe3e4946d5737fc4381817c963fd.p.txt -> #148 @20907
C:\resources\Salmon_numbered_sen_new\c00afe3e4946d5737fc4381817c963fd.p.txt -> #84 @13046
C:\resources\Salmon_numbered_sen_new\f0e39b2582a602c98d266abcbf9e9422.p.txt -> #97 @14988
C:\resources\Salmon_numbered_sen_new\a9ff2614396be82207cd35a2ebbf3af7.p.txt -> #112 @15593
C:\resources\Salmon_numbered_sen_new\5c67853ceaf0f2bd7c9c3e77664e8e6d.p.txt -> #116 @17811
```

Figure 4.3: Example of lines in the result files which are created by dtSearch. Each line consists of an absolute filename string and a byte offset number.

The analysis of dtSearch result files remained mostly unchanged. Each line in the result files consisted of the absolute filename and a byte offset number (see Figure 4.3) referring to documents which matched the query. These documents were opened and read from the byte offset point, up to the tag which marks the end of the document according to the segmentation rule applied in dtSearch (see Section 4.4.1). We changed this tag from **`<p>`** (the tag used in TREC-12) to **`SEN`** in the module that analysed the dtSearch result files. In other words, the system opened a document and started reading and extracting a sentence from the byte offset (verified by the presence of a **`SEN`** tag) and stopped either at the next **`SEN`** or at the end of the document.

Since many sentences contained only a single occurrence of the query term, their ranking by dtSearch, based purely on hit count, was arbitrary. We also assumed that sentences with a higher hit count are not more likely to include definitions, because a proper definition is supposed to avoid using the term it is defining. Therefore, the role of dtSearch was reduced in essence to retrieving any sentence containing the term. However, we set the maximum number of documents (sentences) to be analysed to 1000 to reduce processing time for frequent terms.

We extracted the sentence ordinal number and total sentence number and returned both, as arguments in a list, in the space reserved for the AQUAINT document number (doc id or `Doc_no`) in an asserted clause. This was possible because documents in SOK-i lack identification numbers comparable to those in AQUAINT. Since these numbers appeared always in the two lines after the **`SEN`** tag, their extraction was simple.

When marking the term in the retrieved sentences, we had to match different capitalised versions. The original system generated a version with all the initial letters capitalised and a version which was all in lower case letters, kept the original version, and removed duplicates. We added a version which capitalised only the first letter of the first word in the term. This form is mainly useful when matching a term that consists of two or more words and appears at the beginning of the sentence For example, if the term is ‘response to selection’, the system will generate the following forms:

1. Response To Selection,
2. response to selection (this will be removed because it is identical to the original form),
3. response to selection (original),
4. Response to selection (new form).

The bulk of the changes to the system when adapting it to the current project involved adding Definite-Clause Grammar (DCG) rules to the section of the entity recognition module which specified definition patterns (see Appendix C). We expanded the set of rules in this section from the twenty that were used in TREC-12 for definitions to over 500 in the last experiment.

We disabled all the answer selection components to ensure that there were a substantial number of answers available to study the effectiveness of the pattern extraction technique in each experiment.

Any experimental filtering of the answers was done in the module which generates the final output. This module allowed us to change the output for experimental purposes by extracting selectively different arguments from the final asserted clauses. For example, we were not always interested in printing out the sentence numbers or the context. When generating the final output from these clauses we could also manipulate the information they contained. For example, in one run we calculated at this stage the sentence position by dividing the sentence ordinal number by the total number of sentences in the document. Appendix D shows a sample final output.

4.5 Summary

In this chapter we introduced the design of the DLT question answering system that was used in TREC-11 and TREC-12, focusing on the way it answered definition questions in TREC-12. We then described the pre-processing of the salmon fish documents. This task mainly involved splitting the documents into individual sentences using heuristics before indexing them with dtSearch. We explained how we implemented changes relating to the input, search, retrieval and analysis of the documents, as well as named entity recognition and output. The next chapter presents the experiments that were run using the modified system in different configurations, their results and discussion.