The International Journal Of Engineering And Science (IJES)
||Volume||2 ||Issue|| 8||Pages|| 01-08||2013||
ISSN(e): 2319 – 1813 ISSN(p): 2319 – 1805

Faqcase: A Textual Case-Based Reasoning System

Aminu Bui Muhammad, Abba Almu
Computer Science Unit, Usmanu Danfodiyo University, Sokoto, Nigeria

----------***ABSTRACT***----------
We present FAQCase, a Textual Case-Based Reasoning system for solving Frequently Asked Questions. The system uses an adapted Lesk algorithm to disambiguate senses of query terms and then transform the query with synonyms from WordNet. This is to improve matching at the retrieval phase since, in natural languages; the same information can be expressed using lexically different words. An evaluation of the system shows that there is significant retrieval improvement over a baseline system with no query transformation.

KEYWORDS: *Textual Case-based Reasoning, Word Sense Disambiguation, WordNet.*

Date of Submission: Date13 May 2013, Date of Publication:12 Aug2013,

I. INTRODUCTION

Frequently asked questions (FAQs) are question-answer pairs usually maintained by organizations to help provide first hand answers to recurring questions in the organization. In formulation of FAQs, questions are solved once and stored together with their respective answers, and for any recurrence of any such questions, its already stored answer is reused instead of treating the question as a new problem. For FAQs to effectively serve the purpose for which they are stored, an effective and efficient storage and retrieval technique has to be used for their indexing. There are several methods and techniques that could be used for the storage and retrieval of FAQs which ranges from the use of web pages for display and manual lookup to the use of information retrieval (IR) techniques.

In many organizations, FAQs are stored as web pages and searched manually. This is inefficient and time consuming. Indexing and retrieval using a database can be more efficient; however queries cannot be issued in natural language. Although queries can be issued in natural languages using the IR approach, it may, however, not be very effective as some keywords in the FAQs may have domain-specific importance which cannot be captured based on term frequency alone.

In this work, we investigate the suitability of using case-based reasoning (CBR) approach to the storage and retrieval of FAQs. The CBR problem-solving approach [1] is suited to this task as it advocates the reuse of previous cases to solve new problems. Here the previous cases are the stored FAQs while the problem would be a new question. It, also, allows for the integration of more knowledge during the case comparison stage to improve retrieval performance. Accordingly, we use WordNet to expand user query with synonyms to improve retrieval matching with the stored FAQs. The rest of this paper is organized as follows. The next section reviews the process of developing textual CBR systems. We present the application (FAQCase) in section 3 while experimental setup and results are presented in section 4. The paper ends with conclusion and future work in section 5.

II. TEXTUAL CBR SYSTEMS AND OVERVIEW OF PREVIOUS WORK

CBR is a branch of Artificial Intelligence (AI) that solves problems based on *experience* (i.e. previously encountered problems and their solutions). According to [2], a case is a "contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner". CBR system can also perform more creative problem-solving, that is, it can be applied to solve problems that are quite different from its prior experience. The notion of CBR is based on two world tenets, first, the world is regular; thus, similar problems have similar solutions, and second, the problems an entity encounters tend to recur [3]. These two tenets make it worthwhile to retain solved problems and their solutions so that when such problems recur, their respective solutions can be reused repeatedly.

The CBR problem-solving cycle in figure 1 typically consists of the following four stages [1]:
Retrieve: This involves retrieving cases from memory that relevant to solving problem case.
Reuse: This involves reusing the retrieved cases in order to try and solve the current problem.
Revise: This involves revising the proposed solution and adapting it, if necessary.
Retain: This involves retaining the final solution as a new experience.

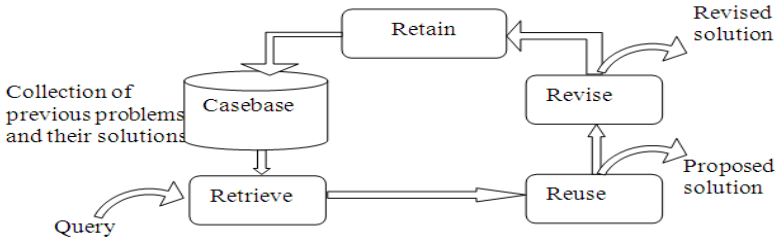

Figure 1: The CBR Cycle

In practice, many of the experiences that a CBR system is to reason with, are collected and stored as textual documents, for example, incidence reports, Frequently Asked Questions (FAQs), Legal cases, documentations and manuals of equipments [4]. This gives rise to textual CBR as a subfield of the standard CBR. Textual CBR (TCBR) can be viewed as a specialised CBR in which the focus is on cases that are in unstructured natural language format, it also allows interaction with the user in the same format. Such flexibility of TCBR systems adds the additional challenge of dealing with free text [5]. Such challenges include the following:

- Mapping text into a structured representation
- Indexing of textual cases to allow fast, efficient and effective comparison/ retrieval
- Measuring similarity between two pieces of text using their grammatical/semantic information as well as specialised knowledge which varies from one domain to another
- Adaptation of retrieved similar textual cases to solve a new problem

Building a TCBR system involves knowledge acquisition process, in terms of the knowledge container model introduced by Richter [4], this includes:

(a) The collection of cases;
(b) The definition of index vocabulary for cases;
(c) The construction of similarity measure; and
(d) The specification of adaptation knowledge.

Taking these into account, we discuss the process of developing textual CBR systems as follows:

2.1 Case Acquisition

This stage in developing a TCBR system involves collecting solved problems with their solutions on which the system is to base its judgement when solving a new problem, i.e. the *experience*. Ideally it is assumed that these cases already exist either in structured or unstructured format.

2.2 Indexing Vocabulary

In TCBR, there is a need to extract vocabulary of the domain from the textual description of cases (documents). Usually the vocabulary is not just independent keywords or terms but may contain phrases and expressions, e.g. *coursework submission*. Each case can then be described based on the *features* (i.e. keywords, terms, expressions etc.) that are in the vocabulary. Such features may be identified manually by classifying the documents based on some properties with the help of the domain expert or, automatically by allowing the machine to analyse the documents and identify the keywords or terms that are indicative of discriminating different cases described by the documents. For example, the keyword *coursework* may be a good feature in discriminating coursework-related FAQs from others. This stage is crucial because only the keywords/features in the vocabulary can be used to represent the case-base and problem case to be solved, hence only these features can be used to compare cases to a problem.

2.3 Case Representation

Upon identification of the appropriate vocabulary to be used to represent cases in a domain, the next step is to use an appropriate model to represent cases. The manner in which cases are represented at this stage can affect the manner in which similarity could be assessed during retrieval [6]. The basic model for document representation in Information Retrieval (IR), i.e. *the vector space model* [7] can also be used to represent cases in TCBR. However, there are other approaches to case representation which include *frame-based and object-oriented* representations; both uses data modelling approach of object-oriented paradigm which include *inheritance principles (i.e. is-a and is-part-of* relations). They are suitable for complex domains where cases with different structure occur [6]. Other advanced approaches such as hierarchical case representation, generalised cases, cases in case-based design and cases in case-based planning have been investigated.

2.4 Similarity Measures*:*

Similarity measure provides a means of quantifying similarity between cases and a problem during retrieval. It is a function that computes the degree of similarity between two vectors. In TCBR, a number of similarity measures could be used in order to capture similarity which is not limited to term occurrence. Cases may be similar based on term occurrence or based on meaning, this makes it important to consider both the similarity based on term occurrence (statistical similarity) and similarity based on meaning (semantic similarity) when comparing cases.

Statistical similarity: This is the measure of term matches between cases, i.e. the more the common terms between the cases, the higher the similarity. There are a number of methods that can be used to measure statistical similarity between cases. These include Inner Product and Cosine Similarity measures [8]. Statistical similarity may suffer from problem of *ambiguity*, where the set of terms between two cases may be highly similar, but the actual meaning of both differs significantly, and *paraphrase* problem, where the same meaning may be expressed using different terms.

Semantic Similarity*:* Semantic similarity is the measure of meaning between to documents/cases. Thus, it can reduce the problems of ambiguity and paraphrase. An efficient and effective strategy of capturing semantic similarity between words is by the use of *synonym* relationship [9].

2.5 Evaluation in Textual CBR

TCBR systems often comprise of multiple aspects (e.g. coverage similarity component, semantic similarity component etc), thus, evaluation has to be designed to measure the performance of the component which is the focus of the research. According to Bruninghaus and Ashley [10], there are three major issues to be considered when evaluating a TCBR system, these are: *What performance measure should be chosen? What should the performance be compared to?* and *What to focus the evaluation on?*

2.6 Previous Work

One of the TCBR systems closely related to our work is the FAQFinder [9]. The system uses a part of a usenet FAQs archive as cases. The FAQFinder system uses a combination of statistical and semantic techniques in its similarity computation between new problem and stored FAQs. The computation of semantic similarity and the entire retrieval matching in FAQFinder was however limited to the question part of the FAQs. Considering the question part of FAQs is generally shorter than the answer part. In this work, we extend the matching to include the answer part of FAQs.

III. THE FAQCASE SYSTEM

The system consists of three basic components namely the *Interface*, the *Search module* and the *Semantic module* as illustrated by Figure 1.

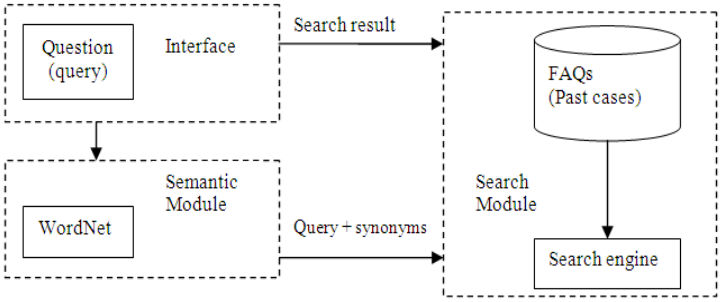

Figure 1: Components of the FAQCase

3.1 The Interface

The interface is the window through which the user interacts with the system, in IR, there are some standard approaches of designing user interface in order to enhance usability of the system. Similarly, in order to facilitate the information seeking process in TCBR, a number of user interface design principles have to be considered. These include provision of informative feedback on the process, relationships between objects (i.e. query and retrieved documents), easy reversal of actions, internal locus of control (i.e. give user control over system where appropriate), reduce working memory (i.e. keep track of progress, save result etc) and alternative interfaces for different users (i.e. novice, expert etc). A typical best match retrieval system (such as TCBR system) consists of *query input, search button, retrieved documents list and document display* [9]. The query input is a text field through which the user sends query to the system. A good query input text field allows simple editing of query. Furthermore, the size of the input area affects the length of query users will type, therefore, it should be as wide as possible to attract richer queries from users. The search button triggers the indexing/search process when clicked. The retrieved documents list display retrieved documents deemed to be relevant to the user query. Finally, the document display area displays the content of a retrieved document.
In FAQCase, the aforementioned design is adopted with extension to incorporate other components. The added components are: *display area* and *user word sense disambiguation pane.* The display area displays synonym sets (synsets) extracted from WordNet and new query formulated using the synsets. This allows the user to see, for each query word, the synset deemed to be of the right sense of the word, and if not satisfied with the systems judgement, the user can change the word sense. The user word sense disambiguation pane contains all the query terms found in WordNet dictionary and the senses of each term. This gives the user the facility to change the sense of word and repeat the search.

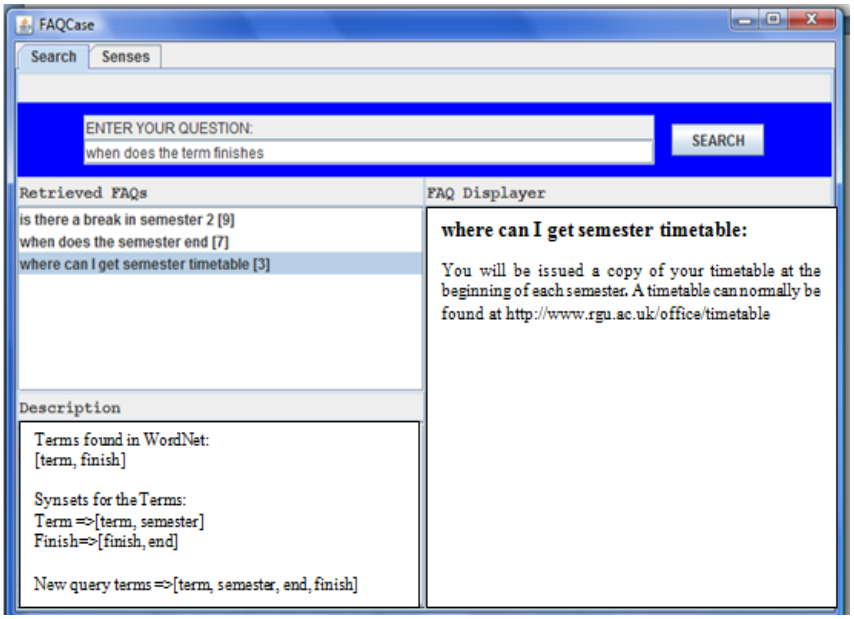

Figure 2: Screenshot of the proposed system (FAQCase)

3.2 The Search Module

This consists of a search engine implemented based on standard IR techniques. On running the application, the search engine generates an inverted file of the stored FAQs after performing a sequence of text pre-processing operations on them. An inverted file is an index structure storing a mapping from words to their locations in a document or a set of documents. We apply the following text pre-processing operations on the collection as well as queries.

Lexical Analysis: At this stage the FAQs were tokenized (i.e. broken down into discrete tokens). These tokens are words, numbers, and punctuations. Here, we adopt the simple approach of ignoring all numbers and punctuation and use only case-insensitive unbroken strings of alphabetic characters as tokens. The resulting tokens from this operation are then passed to the subsequent operation of stop-word removal.

Stop-words Removal: In this stage, language dependent high-frequency words called stop-words are removed from the input tokens. Example of such words are function words: "a", "the", "in", "to"; pronouns: "I", "he", "she", "it". Stop-words removal is generally preferable even though it may give rise to other issues, for example the question "is grade 6 equivalent to A", the token "A" will be reduced to its lower case equivalent "a" during lexical analysis and then considered to be a stop-word at this stage and therefore removed. But then, the "a" in this example is not the same as the function word "a".

STEMMING: Stemming is the process of reducing a token to its "root" form, this reduce the number of distinct terms which results in having smaller index size. It also helps to identify words with similar meaning which enhances recall.

After performing text pre-processing operations, inverted index of tf-idf weighted term-document vectors is generated. We use cosine similarity metric to compute relevance of a given query q to any of the indexed FAQs d_j as follows:

$$CoSim(d_j, q) = \frac{\sum_{i=1}^{t}(d_{ij} \cdot q_i)}{\sqrt{\sum_{i=1}^{t} d_{ij}^2 \cdot \sum_{i=1}^{t} q_i^2}} = \frac{\sum_{i=1}^{t}(w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}} \quad (1)$$

Where w_{ij} and w_{iq} are the weights of the i^{th} term in the document/case and query/problem vectors respectively.

3.3 The Semantic Module

This module uses WordNet [11], to transform query terms with synonyms. This is to improve matching with the indexed FAQs since, in natural languages; the same information can be expressed using lexically different words. The module begins with word sense disambiguation (WSD) in order to identify the right synonyms to be extracted from WordNet. We propose a variant of Lesk algorithm [12] for this purpose (see below). Preliminary evaluation of the algorithm shows that it is as effective as the original Lesk algorithm and computationally less expensive as the original Lesk algorithm.

3.3.1 The WSD Algorithm

WordNet is a machine readable lexical database. It can also be seen as a dictionary, but, unlike most dictionaries, WordNet contains only open-class words (i.e. nouns, verbs adjectives and adverbs), it does not contain any other word which is not in this category. Words' part-of-speech (PoS) have to be identified before being passed to WordNet.We use QTAG [13] to assign PoS to words. The basic relationship between words in WordNet is the Synonym relation called *Synset*. Words in the same synset are synonymous in a particular *sense*. Word sense is the meaning a word can take depending how it is used. For example the word "*bank*" could mean a financial institution in one sense and a river bank in another sense. Each synset of a word contain one or more words including the word itself and has a *gloss* associated with it. A gloss for a word sense is the definition of the word in that particular sense and typically includes example sentence(s). For instance, one of the synsets of 'bank' is {depository financial institution, bank, banking concern, banking company} and its gloss is (a financial institution that accepts deposits and channels the money into lending activities; "he cashed a check at the bank"; "that bank holds the mortgage on my home"). In this work, we use WordNet 3.0 for the implementation of WSD component of FAQCase. This version contains about 152,000 different words which have a total of about 203,000 different senses. It has over 115,000 synsets, of these synsets about 80,000 are noun synsets, 13,500 are verb synsets, 18,500 are adjective synsets and 3,700 are adverb synsets.

Lesk [12] proposes WSD algorithm known as the Lesk Algorithm. The basic idea behind the Lesk Algorithm is based on the assumption that words that appeared in the same context are likely to have overlapping words in their dictionary definitions. When comparing the dictionary definitions or glosses of two words, the Lesk algorithm counts up the number of tokens that occur in both glosses and assigns a *score* equal to the number of words thus matched to this gloss–pair. In our implementation of the algorithm score of matched gloss-pair is not the count of overlapped words between the glosses rather cosine similarity between them. The following is an outline of the implemented Lesk Algorithm:

1: Function diambiguateAllWords(words) returns bestSense
2: **for all** w_t in input **do** {w_t is known to WordNet and not found in stoplist}
3: bestSense_t = diambiguateSingleWord(w_t)
4: **end for**
5: **end procedure**
6: **function** disambiguateSingleWord(w_t) return sense s
7: **for all** s_{ti} of target word w_t do {s_{ti} is the i^{th} sense of the target word w_t}
8: $\text{score}_i = 0$
9: **for** $j = t\text{-}c_l$ to $t+c_r$ **do**
10: **if** $j = t$ then
11: next j
12: **end if**
13: **for all** s_{jk} of w_j **do**{s_{jk} is the k^{th} sense of word w_j}
14: tempScore = relatedness(s_{ti}, s_{jk})
15: **end for**
16: bestScore = max tempscore
17: **if** bestScore > threshold then
18: $\text{Score}_i = \text{score}_i$ + bestScore
19: **end if**
20: **end for**
21: **end for**
22: **return** sense_i scorei > score_j for all j, where j not equal i
23: **end function**

The function disambiguateAllWords takes in the words to be disambiguated and return their right senses. At the loop (2:) the algorithm iterates over all the words and pass each to the function disambiguateSingleWord. The function disambiguateSingleWord disambiguates the sense of a word. At the loop (7:) , it iterates over all the senses of the word, at the loop (9:), it takes a window of the target word's neighbours whose glosses will be used to disambiguate the sense of the target word. In this implementation of the algorithm, all the query terms were considered to be neighbours of one another, this is because user queries are generally short. At the loop (13:) the algorithm iterates over senses of a neighbour word (w_j) and calculate the relatedness of each of its glosses with the glosses of the target word (the word to be disambiguated).

After disambiguating senses of query terms, we can now get the synonyms (synsets) of each term. User queries are then expanded with the synsets.

For example, the questions

A "*I was unable to attend exam due to illness – what should I do?"* and

B *"I missed examination because of sickness - any help?"*

The two questions are supposed to have exactly the same meaning, but matching the two questions as they are now will give zero-similarity for they do not have any common keyword.

Keywords of A = [was, unable, attend, exam, due, illness, what, should]

Keywords of B = [missed, examination, because, sickness, any, help]

In this research, WordNet was used to get the appropriate synset for query terms in order to expand the query, in the above example, if A is the stored FAQ and B is the new problem, by adding synonyms to B some similarity will arise between the two questions. This can be shown as follows:

Keywords of A = [was, unable, attend, exam, due, illness, what, should]

Keywords of B = [missed, examination (exam), because, sickness (illness) any, help]

The bracketed terms are synonyms of their preceding terms extracted from WordNet. This will lead to FAQ A being retrieved as a relevant case to the problem case B.

IV. EVALUATION

According to Bruninghaus and Ashley [10], there are three major issues to be considered when evaluating a TCBR system, these are: *What performance measure should be chosen? What should the performance be compared to?* and *What to focus the evaluation on?*. Considering this, we chose *Normalised Precision* [14] as performance measure. This compares the precision performance of systems between worst case and best case; it ranges between 1 and 0. Worst case (0) is when the system did not retrieve the targeted answer, best case (1) is when the answer is retrieved and ranked top and anything between 0 and 1 is when the answer is retrieved but not ranked the top. This performance measure has been chosen because there is only one FAQ (if it exists) in the case base which is expected to answer user question/query, thus, the it measures how well the system retrieved and ranked this answer between the system in comparison.

4.1 Dataset Characteristics

The Dataset consists of 22 FAQs from a university domain. The FAQs were obtained from staff members who attend to students' complaints and questions. A total number of 143 different terms were used in forming the test FAQs out of which 127 were found in WordNet, among those found in WordNet, 114 have the sense to be disambiguated and 13 do not have.

4.2 Experimental Design

A total of 22 FAQs of the stored FAQs were rephrased by 3 users, the users were drawn from the university, 1 staff member, 1 research student and a Masters student. Some of the rephrased FAQs contain combination of terms in their original FAQs and other different words while others were formed with completely different words from their original FAQs. Example:
Original FAQ "*I was unable to attend exam due to illness – what should I do?"*
Rephrased FAQ *"I missed examination because of sickness - any help?"*
Performances of two systems were measured, the description of the two systems are as follows:

System A: this is version of FAQCase, the developed application that transforms user query following word sense disambiguation of the query terms (FAQCase with Word Sense Disambiguation and query transformation).

System B: this has everything like System A except it does not perform word sense disambiguation and query transformation (FAQCase without Word Sense Disambiguation and query transformation).
The 3 rephrased versions of the case-base FAQs formed the test FAQs (Test FAQs_1, Test FAQs_2 and Test FAQs_3) for the experiment. The ability of the two systems to recover from the paraphrasing of the FAQs was measured using Normalised Precision.

4.3 Results and Discussion

Table 1 shows performance of the two systems on the rephrased FAQs, reported as average normalised precision across the test FAQs.

System / Dataset	System A	System B
Test FAQs_1	**0.97**	0.77
Test FAQs_2	0.83	0.74
Test FAQs_3	**0.94**	0.60

Table 1: Experimental results reported as average normalized precision

The result shows the proposed system (System A) to outperform a competing approach of using standard IR techniques with no query expansion. The proposed system is better on all the 3 test FAQs. The difference is statistically significant (at 95% confidence level) on test FAQs_1 and test FAQs_3. This can be attributed to the fact that most FAQs from the two datasets were formed with completely different words with their equivalents in the case-base. The result has, therefore, shown that, when user questions are likely to be completely of different words from a FAQ questions likely to answer them then word sense disambiguation and query transformation as proposed in this work are desirable.

V. CONCLUSION AND FUTURE WORK

In this research, FAQCase, a TCBR System has been presented, which uses WordNet lexical database to transform user queries/questions. The query transformation was achieved by using WordNet's glosses and Lesk Algorithm to disambiguate query terms after which, appropriate synonyms of query terms are added to queries. The results show significant retrieval improvement the system (FAQCase) over the one without query transformation. WordNet is a general English database, as such it lacks some domain specific terms. In the problem domain of this work, there are some terms that cannot be found in WordNet such as the term *resit.* Also some terms could be found in WordNet but there usage in a particular could not be in WordNet for instance the term *account* could be found in WordNet, but, *account* in the sense of *email account* could not be found in WordNet. Therefore future work related to this research is on building a domain specific thesaurus for the School of Computing.

REFERENCES

[1]. A. Aamodt and E. Plaza, Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, Artificial Intelligence Communications 7(1), (1994), 39-52.

[2]. J. Kolodner, Case-Based Reasoning (San Mateo, California: Morgan Kaufmann, 1993).

[3]. D.B. Leake , CBR in Context: The Present Future. In D.B. Leake (Ed), Case-Based Reasoning: Experiences, Lesson and Future Direction (Menlo Park: AAAI press, 1996) 3-30

[4]. M. Lenz, Textual CBR and Information Retrieval – A Comparison. In: L. Gierl, and M. Lenz (Ed), Proceedings of sixth German Workshop on Case-based Reasoning Rostock, Germany (Rostock: University of Rostock, 1998)

[5]. M. Lenz, Knowledge sources for textual CBR applications, In M. Lenz and K. Ashley (Ed), Proceedings of the AAAI-98 Workshop on Textual Case-Based Reasoning (Menlo Park, CA: AAAI Press, 1998) 24-29

[6]. R. Bergmann, J. Kolodner and E. Plaza, Representation in case-based reasoning, The Knowledge Engineering Review 00(0), (2005), 1-4

[7]. G. Salton, A. Wong, and C. S. Yang, A Vector Space Model for Automatic Indexing, Communications of the ACM 18(11), (1975) 613–620

[8]. A. Singhal, Modern Information Retrieval: A Brief Overview, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4), (2001) 35–43

[9]. R.D. Burke, K.J. Hammond, V. Kulyukin, S.L. Lytinen, N. Tomuro, and S. Schonberg, Question-Answering from Frequently-Asked Question Files: Experiences with the FAQ-Finder System, AI Magazine, 18(2), (1997) 57-66

[10]. S. Brüninghaus and K.D. Ashley, Evaluation of Textual CBR Approaches, Proc. of the AAAI-98 Workshop on Textual Case-Based Reasoning, Madison, WI.: USA, 1998.

[11]. C. Fellbaum, WordNet: An Electronic Lexical Database (London: The MIT press, 1998)

[12]. M. Lesk, Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to tell a pine cone from an ice cream cone, Proc. of the 5th international conference on systems documentation, New York, NY,: USA, 1986, 24-26.

[13]. The University of Birmingham. QTAG. [homepage on the internet]. Birmingham: c.1994-2003 [cited 2010 June 11], available from: http://www.english.bham.ac.uk/staff/omason/software/qtag.html

[14]. C. J. Van Rijsbergen, Information Retrieval, (Newton, MA: Butterworth-Heinemann, 1979)