RULE GENERATION BASED ON STRUCTURAL CLUSTERING FOR AUTOMATIC QUESTION ANSWERING

BY

SONG SHEN

Thesis submitted in fulfillment of the requirements
for the degree of
Master of Science

December 2009

Acknowledgment

I would like to take this opportunity to convey my sincere thanks and deepest gratitude to my main supervisor, Dr. Cheah Yu-N and my co-supervisor, Prof. Tang Enya Kong, for their dedication, support, encouragement, patience and valuable guidance provided to me during the preparation of this thesis.

Moreover, I would like to convey my appreciation to Health Informatics Research Group, School of Computer Sciences, Institute of Postgraduate Studies, the university library for their help and support.

Finally and most important of all, I would like to express my most sincere and warmest gratitude to my mother Dong Feng, my father Song JingSheng and my husband Usman Sarwar. Thanks for their love and care. They have always encouraged me, believed in me, and supported me when I needed it.

TABLE OF CONTENTS

CHAPTER 3 - METHODOLOGY

LIST OF TABLES

LIST OF FIGURES

PENJANAAN PETUA BERDASARKAN PENGELOMPOKAN STRUKTUR UNTUK PENJAWABAN SOALAN AUTOMATIK

ABSTRAK

Dalam kaedah berdasar-aturan untuk penyelidikan soal-jawab (QA), teknik pembelajaran aturan tipikal adalah didasarkan pada pertindihan corak dan maklumat leksikal. Hal ini biasanya terhasil dalam aturan yang boleh memerlukan interpretasi lanjut dan aturan yang mungkin berlebihan. Bagi menangani isu ini, suatu algoritma penjanaan aturan berstruktur automatik dibangunkan melalui pengklusteran, dan suatu kaedah pengklusteran berasaskan-ayat pusat diolah untuk menjana aturan bagi sistem QA secara automatic.

Metodologi bagi penyelidikan ini melibatkan tiga fasa. Fasa pertama melibatkan prapemprosesan pasangan soal-jawab latihan yang diterbitkan daripada korpus pemahaman bacaan 4 kanak-kanak CBC (Canadian Broadcasting Corporation). Prapemprosesan juga melibatkan tag POS (part-of-speech). Fasa kedua pula melibatkan penjanaan aturan secara automatik, dan tag-POS pasangan QA dikluster berdasarkan keserupaan nombor token POS dan jujukan mereka. Untuk ini, kaedah komputan serupa (similarity computation method) BLEU digunakan. Yang terakhir, fasa ketiga, melibatkan operasi sistem QA yang dikenali sebagai Sistem Soal-Jawab berdasarkan

Penjanaan Aturan Automatik (QASARG). Output daripada sistem ini kemudiannya dinilai.

Keberkesanan QASARG dinilai terhadap sistem berasaskan aturan QA lain, Quarc. Ketepatan QASARG adalah dalam julat 55% hingga 85% bergantung pada jenis soalan, dan secara purata 26.4% lebih tinggi daripada Quarc. Walau bagaimanapun, perlu diambil perhatian bahawa set data ujian yang digunakan untuk menilai QASARG dan Quarc adalah berbeza (QASARG diuji berdasarkan pasangan QA yang diterbitkan daripada bahagian pemahaman bacaan, sedangkan keputusan Quarc adalah berdasarkan keseluruhan bahagian pemahaman bacaan). Namun demikian, keputusan QASARG menunjukkan bahawa keserupaan struktur di antara ayat adalah berguna dalam menjana aturan yang tepat untuk QA secara munasabah dan boleh dipercayai.

RULE GENERATION BASED ON STRUCTURAL CLUSTERING FOR AUTOMATIC QUESTION ANSWERING

ABSTRACT

In rule-based methods for Question-Answering (QA) research, typical rule discovery techniques are based on structural pattern overlapping and lexical information. These usually result in rules that may require further interpretation and rules that may be redundant. To address these issues, an automatic structural rule generation algorithm is presented via clustering, where a center sentence-based clustering method is designed to automatically generate rules for QA systems.

The methodology for this research involves three phases. The first phase involves pre-processing of training question-answer pairs derived from the Canadian Broadcasting Corporation's (CBC) 4 Kids reading comprehension corpus. Pre-processing also involves part-of-speech (POS) tagging. The second phase involves automatic rule generation where the POS-tagged QA pairs are clustered based on the similarity in matching POS tokens and their sequences. For this, the BLEU similarity computation method is employed. The final phase involves the operationalisation of the QA system called Question Answering System based on Automatic Rule Generation (QASARG). The output from this system is then evaluated.

The effectiveness of QASARG was evaluated against another rule-based QA system, Quarc. The accuracy of QASARG is in the range of 55% to 85% depending on the question type, and these are on average 26.4 % higher than those for Quarc. However, it must be noted that the test data sets used to evaluate QASARG and Quarc are different (i.e. QASARG is tested based on question-answer pairs derived from the reading comprehension passage while Quarc's results are based on the entire reading comprehension passages). Nevertheless, the results for QASARG indicate that structural similarities between sentences are useful in generating reliable and reasonably accurate rules for QA systems.

CHAPTER ONE
INTRODUCTION

1.1 From Information Retrieval to Question Answering

Ever since the emergence of human civilization, we have already realized that the proper organization and access to the archives were critical for efficient use of information, i.e. the Sumerians had designated special areas to store clay tablets with cuneiform inscriptions in 3000 B.C. (Singhal & Google Inc., 2001).

Over the centuries, the demand of store and retrieve written information became important. With the invention of computers, people realized that computers could be used for storing and retrieving large amounts of information. The idea of using computers to search for relevant pieces of information was popularized in an article "As We May Think" by Vannevar Bush in 1945 (Bush, 1945). By 1990 several different information retrieval systems had been shown to perform well on small text corpora.

However, with the rapid increase in information nowadays, we are now faced with the problem of retrieving relevant information from various redundant resources such as documents and the Internet. This is largely due to the sheer information overload. Search engines have been proven useful in addressing many keyword-related search initiatives. Nevertheless, their effectiveness lies in the skill of the users to construct the right queries.

To further facilitate the search for information and to improve the user interface, automatic Question Answering (QA) approaches have been developed as a specialized information retrieval domain to allow questions to be posed in natural language (Hagen, Manning & Paul, 2000).

1.2 Question Answering Systems

As a branch in the field of Information Retrieval (IR), the initial purpose of QA systems is to provide a simple natural-language interface to expert systems. Nowadays, QA systems have moved on to become the next generation of search engines, with the capability to retrieve precise answers rather than related links (e.g. Google).

QA systems avoid the need for users to formulate structured queries in order to retrieve a particular piece of information. Another added advantage is that QA systems also have the potential to respond to a user's query in natural language. The rapid development of question answering technologies in recent years leads to an increasing interest on the side of researchers, companies and end users.

Since the first QA systems developed in the 1960s, e.g. BASEBALL (Green et al., 1961), more and more QA systems were implemented under the motivation of the Text Retrieval Conference (TREC) in late 1990s. With systems such as Start (Katz, 1988) and AnswerBus (Zheng, 2002), researchers were trying different methods from different angles to improve the answer retrieval capability of QA systems.

Research on QA systems can be roughly categorized into three areas, information repository, query analysis, and answer matching (i.e. corresponding to the different phases in QA). In the area of information repository, the core task is to preselect the relevant documents or texts for extracting answer candidates of the input questions. Meanwhile, the information from the input question is the most important clue in answer retrieval. Hence, the other research area is targeted on query analysis. Answer matching is based on the efforts of the previous two phases and also on an effective answer retrieval method. The more effective the pre-selection of documents as well as the methods of query analysis and answer retrieval are, the higher the possibility of retrieving the matched answer to the input question.

Generally, the development of QA systems requires solid foundations both in the areas of software engineering and Natural Language Processing (NLP), and therefore involves a wide range of techniques (Voorhees, 2001):

1. Information repository: traditional document retrieval and information extraction techniques are exploited to pre-select the documents and the text of the documents which possibly contain the candidate answers of the question, as well as named entities in the question. In searching the information repository, QA systems can be divided into two categories, i.e. closed-domain (which deals with questions under a specific domain, e.g. medicine), and open-domain (which deals with questions in any domain, and relies on a general ontology).
2. Query analysis: regular expression or machine-learning techniques are exploited to classify the questions according to the type of expected answers.

3. Answer matching: keywords, different parsers, and logical proof tools are commonly utilized to help retrieve candidate answers for the input question (Hirschman & Gaizauskas, 2001). The choice of answer matching technique categorizes QA systems further into inference-based, NLP tools-based, cooperative, or rule-based QA system. Many of these QA systems consider both semantic and syntactic factors of the question and answer sentences.

1.3 Rule-Based QA system

Traditionally, rule-based approaches have been employed in QA systems for the matching mechanism in view that it is simple, efficient and effective. Generally, it involves exploring the relationship among patterns within one sentence (question or answer sentence) with the help of NLP tools or specific weighted keywords matching techniques.

Rules can be generated manually. E.g. in Quarc (Riloff & Thelen, 2000), it contains a list of hand-crafted rules for each type of question. The rules covered keyword matching and some lexical clues. On the other hand, some rule-based QA systems involve automatically generated rules. However, most of these rules were generated semi-automatically. E.g. in AnswerFinder (Molla & Zaanen, 2005), based on the overlapping of each question-answer sentence pair, graph rules were generated for each specific question/answer pair according the heuristics. In another system called TextRoller (Soubbotin & Soubbotin, 2001), a complex hierarchy of indicative pattern rules was applied on surface strings in manual method at first and extended the patterns infinitely by inferring new patterns while studying the corpora. For the semi-automatic learning of rules for QA systems, the typical learning method is based on pattern overlapping and lexical information in general. The QA rules are still very much dependent on humans understanding and intervention. Furthermore, these rules are very specific to certain sets of training data (question/answer pairs).

1.4 Problem Statement

As mentioned above, various initiatives are on-going to realize fully automatic rule learning as most of the current automatic rule learning methods still partially depend on human understanding. Moreover, most of the rules in these rule-based QA systems are to detect similar pattern relationships between questions and corresponding answers. As a result, these rule-based QA systems are quite specific to its training resources. Meanwhile, there are some rule-based QA systems only identify sentences which contain the answers rather than directly answer the questions, e.g. Quarc.

1. 5 Research Objectives

According to the structural relationship between questions and their corresponding answers, and also with the purpose of obtaining more general rules for QA systems, the objectives of this research are:

- To define a clustering algorithm to generalize question and answer sentence structures resulting in a fully automatic rule generation mechanism for QA without the need of human understanding.

- To assess whether the automatic rule clustering algorithm and structural information are able to improve the accuracy of the QA system.

1.6 Research Scope

In this research, the training corpus is limited to reading comprehension passages with answers that have been reworded based on certain sentences in the corpus. This is to ensure that the answer sentences directly answer the question (as opposed to taking a sentence verbatim). The automatically generated rules are limited to structural rules, i.e. the rules indicate that questions with a particular structural pattern will require an answer of a particular structural pattern. Also, instead of other popular similarity measurement for clustering, the BLEU (Papineni et al., 2002) mechanism is employed to measure the similarity distance between any two sentences, which used to be utilized in machine translation. Structural information is considered useful for certain research fields, i.e. plagiarism detection, and it could be further emphasized in QA. In our research, we only consider the structural relations between sentences to observe the efficiency of structural information for QA.

1.7 Contribution

In this thesis, a clustering algorithm was successfully designed to generate structural rules for QA system without the need for human understanding. The clustering algorithm (inspired by QT clustering (Heyer, Kruglyah & Yooseph, 1999)) considered the similarity of word order (or structural sequence) between questions and answers. Therefore, instead of considering the lexical relationship between sentences, the sentences (questions or answers) were regarded as a sequence of POS tokens, which was similar to the gene sequence concept used with QT clustering.

Generally, the clustering techniques in other fields of NLP require the feature weights of each sentence, which means that a feature weight is assigned to each sentence. Based on those assigned feature weights, the similarity between two sentences is measured by comparing the similarity between the two feature weights. Different from this similarity measurement, in this thesis, the similarity between two sentences was determined by calculating the structural similarity between any two sentences, which means that the similarity distance was assigned to a pair of sentences, rather than assigning a feature weight to each single sentence. In our proposed rule generation algorithm, the BLEU method (Papineni et al., 2002) was utilized for measuring similarity distances between any two (question or answer) sentences.

Besides the clustering algorithm, a QA system called Question Answering System via Automatic Rule Generation (QASARG) was developed to allow answers to be returned as output based on the rules that were generated by the clustering algorithm. This also allowed the QA system generally, and the rules particularly, to be evaluated.

1.8 Thesis Outline

This thesis is presented in six chapters. The following is an overview of each chapter:

- Chapter 1 (Introduction): This chapter gives a brief introduction and background of QA systems as an important branch of IR. Meanwhile, the research objectives and contributions are also presented.
- Chapter 2 (Literature Review): This chapter presents a survey of QA systems, especially rule-based QA systems. Some existing QA systems based on other answer extractions methods are described in this chapter as well. Lastly, clustering in NLP is also presented in brief.
- Chapter 3 (Methodology): This chapter outlines the research methodology. The methodology is presented in three phases: (1) pre-processing of training question-answer pairs, (2) designing of an algorithm for automatic rule generation via center sentence-based clustering, and (3) implementing a QA system to assess the rules generated in the second phase.
- Chapter 4 (Implementation): This chapter presents the implementation of the center sentence-based clustering algorithm for automatic QA rules generation. A simple example is shown to explain how the rules are generated via the clustering method.
- Chapter 5 (Evaluation): This chapter describes an evaluation of the generated rules based on different conditions (i.e. different Q-threshold, A-threshold, and N-gram values). For each question type, the suitable Q-threshold(s) and A-threshold(s), and also the best N-gram are decided based on the analysis of results for each type of question. Meanwhile, a comparison between QASARG and Quarc is also presented.
- Chapter 6 (Conclusion): This chapter presents a summary of this research work, and re-visits the research objectives and contributions. An outline of future work is also discussed.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

In this chapter, existing Question Answering (QA) systems and answer extraction techniques are surveyed. In line with the research problems mentioned in Chapter 1, rule-based QA systems are highlighted in this survey. However, QA systems based on other methods are also reviewed. Moreover, clustering methods in Natural Language Processing are also reviewed in view that one of the contributions in this research is to develop an algorithm for automatic rule generation for QA based on clustering.

2.2 Rule-based Question Answering System

Traditionally, rule-based approaches have been employed in QA systems for the answer matching mechanism in view that it is simple, efficient and effective. Initially, the rule-based approach for QA involves the manual generation of rules. Subsequently, automatic learning algorithm of QA rules is carried out as well. However, most of the rules are generated semi-automatically, requiring intervention from humans in answer matching.

2.2.1 Quarc

Quarc (Riloff & Thelen, 2000) is a heuristic rule-based QA system focusing on reading comprehension passages. The heuristic rules that are derived look for both lexical and semantic clues in the question and the passage/story. There are five sets of rules according to the interrogative types (WHAT, WHO, WHY, WHERE and WHEN). Figure 2.1 shows an example of WHAT-type rules in Quarc. From Figure 2.1, Rule 1 is the generic word matching function shared by all question types. Rule 2 rewards sentences that contain a date expression if the question contains a month of the year. Rule 3 addresses these questions by rewarding sentences that contain certain words. Rule 4 looks for words associated with names in both the question and the sentence. Rule 5 recognizes questions that contain phrases such as “name of <x>”.

```
1. Score(S) += WordMatch(Q,S)

2. If contains(Q,MONTH) and contains(S,{ today,
   yesterday, tomorrow, last night})

   Then Score(S) += clue

3. If contains (Q, kind) and contains(S, {call, from})

   Then Score(S) += good_due

4. If contains(Q,narne) and contains(S, { name, call, known} )

   Then Score += slam_dunk

5. If contains(Q,name+PP) and contains(S,PROPER_NOUN)
   and contains(PROPER_NOUN,head(PP))

   Then Score(S) += slam_dunk
```

Figure 2.1: An example of manual rules in Quarc (Riloff & Thelen, 2000)

Given a question and a passage, Quarc parses the question and all of the sentences in the passage using the Sundance partial parser. Most of the syntactic analysis is not used. The rules are applied to each sentence in the passage, as well as the title of the passage, with the exception that the title is not considered for WHY questions. Each rule awards a certain number of points to a sentence. After all of the rules have been applied, the sentence that obtains the highest score is deemed to contain the answer.

2.2.2 TextRoller

In the case of TextRoller (Soubbotin & Soubbotin, 2001), it uses not only keywords, but also a complex hierarchy of indicative pattern rules on surface strings for choosing and arranging candidate answers. The definition of indicative patterns is totally heuristic and inductive. The indicative patterns used by TextRoller are sequences or combinations of certain string elements. At the initial stage, the indicative pattern lists are accumulated based on expressions that can be interpreted as answers to the questions of a definite type. The system studies texts systematically with the purpose of identifying expressions that may serve as models for answer patterns. The library of patterns can never be completed. Thus, the system can accumulate the knowledge on 'typical' combination and correlations of strings.

A pattern may include a constant part and a variable part. The latter can be represented by a query term or even an unknown term. Usually, patterns with more sophisticated internal structure are more indicative of the answer. The combinations of element for patterns are also used. There are six basic definition patterns which can answer not only definition question but also WHO, WHERE, and other question types. The example of indicative patterns is shown in Figure 2.2.

1. <A; is/are; [a/an/the]; X>
<X; is/are; [a/an/the]; A>
Example: "Michigan's state flower is the apple blossom".

2. <A; comma; [a/an/the]; X; [comma/period]>
<X; comma; [a/an/the]; A; [comma/period]>
Example: "Moulin Rouge, a cabaret ".

3. <A; [comma]; or; X; [comma]>
Example: "shaman, or tribal magician."

4. <A; [comma]; [also] called; X [comma]>
< X; [comma]; [also] called; A [comma]>
<X; is called; A>
<A; is called; X>
Example: "naturally occurring gas called methane".

5. <X, dash; A; [dash] A; dash; X; [dash]>
Example: "nepotism - hiring relatives for the better jobs".

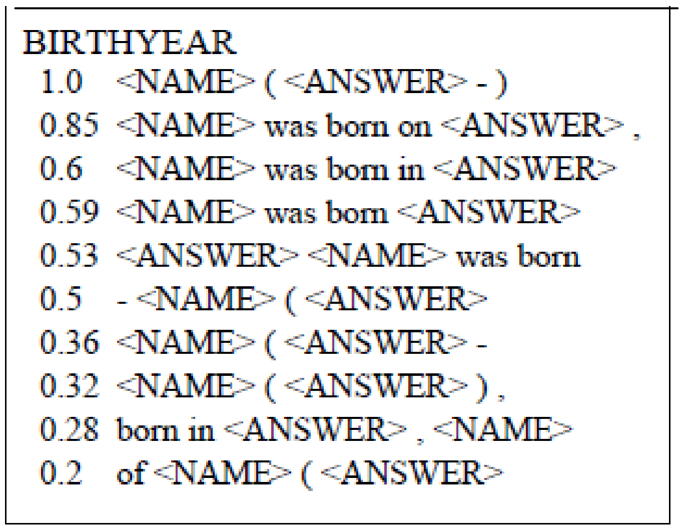
6. <X; parenthesis-; A; parenthesis >
<A; parenthesis; X; parenthesis >
Example: "myopia (nearsightedness)".

Figure 2.2: An example of indicative patterns (Soubbotin & Soubbotin, 2001)

In TextRoller, questions are analyzed in terms of question types. Using specific question words as query terms (known as primary keywords) ensure in most cases that the question subject is addressed in the source passages. In some question categories, primary words do not convey the question subject completely, requiring secondary searching terms. Query expansion may also be required in certain cases. The retrieved passages are cut into 50-byte snippets which are around the query words, or other question words. All the snippets are analyzed to identify patterns that are indicative of a potential answer based on a confidence score.

2.2.3 Webclopedia

Webclopedia (Papineni et al., 2002) is a question answering system using manually learned patterns. To classify the QA types, knowledge about language and about the world are both involved in improving the results. Patterns are learned manually from the web. Altavista is used to return 1000 relevant documents, and only sentences containing both the question terms and answer terms are retained. For each document, the sentences containing more words and phrases that overlap with the question and its expanded query words are extracted and ranked. Webclopedia classifies desired answers by their semantic types, using the approx. 140 classes called Qtargets. In Webclopedia, there are 5 types of Qtargets, i.e. abstract, semantic, syntactic, role, and slot. The example of Webclopedia Typology is shown in Figure 2.3.



BIRTHYEAR

1.0	<NAME> (<ANSWER> -)
0.85	<NAME> was born on <ANSWER> ,
0.6	<NAME> was born in <ANSWER>
0.59	<NAME> was born <ANSWER>
0.53	<ANSWER> <NAME> was born
0.5	- <NAME> (<ANSWER>
0.36	<NAME> (<ANSWER> -
0.32	<NAME> (<ANSWER>) ,
0.28	born in <ANSWER> , <NAME>
0.2	of <NAME> (<ANSWER>

Figure 2.3: Example patterns on BIRTHYEAR (Papineni et al., 2002)

The techniques involved in the architecture of Webclopedia, i.e. question analysis, document/passage retrieval, passage analysis for matching against the question, are adapted from current standard QA systems. To the input question, CONTEX is used to obtain a semantic representation of the questions and answer candidates (Heyer, Kruglyak & Yooseph, 1999). Then, these phrases/words are assigned significance scores according to the frequency of their type in the question corpus. Following this, IdentiFinder is used to isolate and classify names in texts. According to the parsing result from IdentiFinder, WordNet synsets are used for query expansion by means of a series of Boolean queries. After query expansion, relevant documents are retrieved by the search engine called MG (Ulf, Hovy & Lin, 2002). According to similarity between patterns of answer sentences and the query, the most relevant answer passages are ranked for the last phase (answer matching). In answer matching, the parse trees of the relevant answers are compared with the parse tree of the original question. Thereby, the most suitable answer candidate(s) are extracted according to the ranking (Hovy et al., 2000).

2.2.4 AnswerFinder

This is a QA system using logical graph rules which is learnt semi-automatically. The rule learning method is based on the translation of the logical forms of questions and answers into graph form (Witten, Moffat & Bell, 1994). This graph rule learning method is quite straightforward as shown in Figure 2.4. Rules learnt with this algorithm are very specific to question-answer pairs. Hence, the rules need to be generalized. Each generalized rule is weighted according to its ability to detect the correct answer in the training corpus.

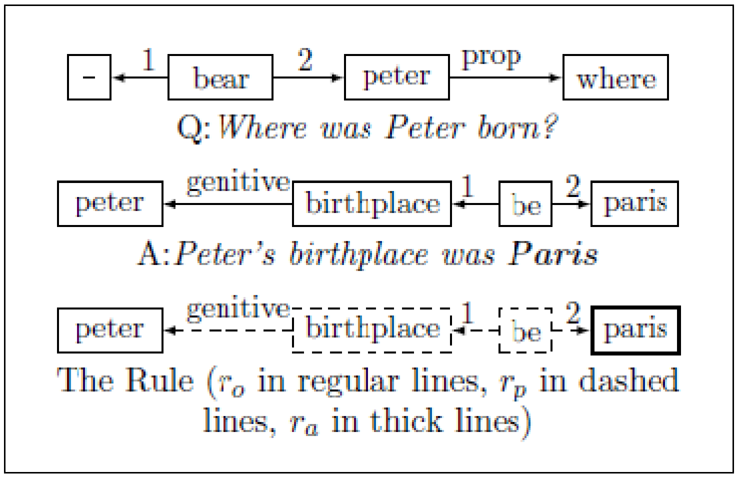

–
1
bear
2
peter
prop
where
Q: Where was Peter born?
peter
genitive
birthplace
1
be
2
paris
A: Peter's birthplace was Paris
peter
genitive
birthplace
1
be
2
paris
The Rule (r_o in regular lines, r_p in dashed lines, r_a in thick lines)

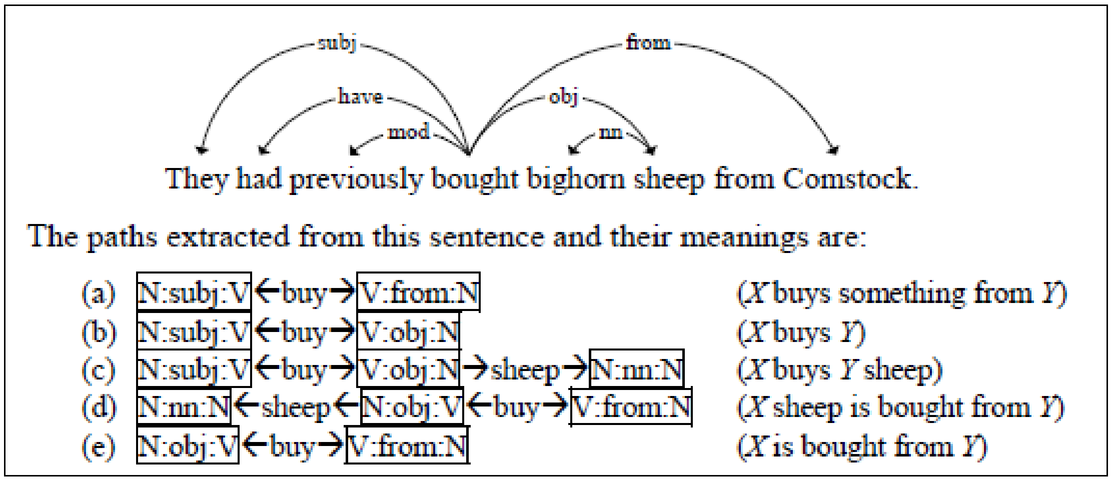
Figure 2.4: A logical graph rule (Witten, Moffat & Bell, 1994)

In AnswerFinder, the process of finding the answer iterates over all the rules according to the given question *q* with graph *Q* and a sentence *s* with graph *S*. A rule *r* triggers if the overlap component of the rule is a sub graph of *Q*. After generating the expanded graph, the overlap is computed between this expanded graph and that of the answer sentence. The result of overlapping between question and answer sentences determines whether the answer sentences can be extracted as the final answer candidate.

2.2.5 DIRT

Inference rules are used in DIRT (Ravichandran & Hovy, 2002). However, the inference rules are learned through an unsupervised algorithm instead of a traditional manual one. The inference rules, learned by DIRT, are to find similar words by means of detecting similar paths in dependency trees. If two paths tend to occur in similar contexts, the meanings of the paths tend to be similar.

The dependency trees are generated by Minipar (Lin & Pantel, 2001), which is a principle-based English parser and the lexicon of it is derived from WordNet (Berwick, Abney & Tenny, 1991). Additional nodes and links are created dynamically to represent subcategories of verbs. In the dependency trees generated by Minipar, a simple transformation is applied to connect the prepositional complement to the words modified by the preposition. Therefore, each link between two words in a dependency tree represents a direct semantic relationship. An example of extracted paths is shown in Figure 2.5.


subj
have
mod
from
obj
nn
They had previously bought bighorn sheep from Comstock.
The paths extracted from this sentence and their meanings are:
(a) N:subj:V←buy→V:from:N (X buys something from Y)
(b) N:subj:V←buy→V:obj:N (X buys Y)
(c) N:subj:V←buy→V:obj:N→sheep→N:nn:N (X buys Y sheep)
(d) N:nn:N←sheep←N:obj:V←buy→V:from:N (X sheep is bought from Y)
(e) N:obj:V←buy→V:from:N (X is bought from Y)
Figure 2.5: Example of extracted paths (Ravichandran & Hovy, 2002)

The path is a binary relation between two entities. A path begins and ends with two dependency relations, which are called two slots and in charge of left-hand side and right-hand side paths respectively. The similarity between a pair of paths is defined as the geometric average of the similarities of their slots (Miller et al., 1990).

2.3 Other QA Systems

Besides the rule-based approach, other approaches have also been utilized in QA systems. Here, some of these other approaches are discussed.

2.3.1 WEBCOOP

The concept of cooperative answer is proposed by Grice (Szpektor & Dagan, 2009) in the 1970s. A cooperative answer is an answer that should be correct, non-misleading, and answers a query. In order to measure the cooperative performance of an information system, there are maxims that describe fundamental properties of cooperative behavior (Gallaire, 1978), which include quality, quantity. QA systems eventually adopted the concept of cooperative answer. One such system is WEBCOOP (Gaasterland, Godfrey & Minker, 1992).

WEBCOOP (Benamara, 2004) is a QA system that provides intelligent cooperative responses to Web queries. This QA system integrates knowledge representation and advanced reasoning procedures to assist answer extraction instead of utilizing NLP tools frequently, i.e. processing queries or generating responses. Meanwhile, the inclusion of answer justification features help to provide a wider range of relative information compared to inference-based system. The architecture of WEBCOOP is shown in Figure 2.6.

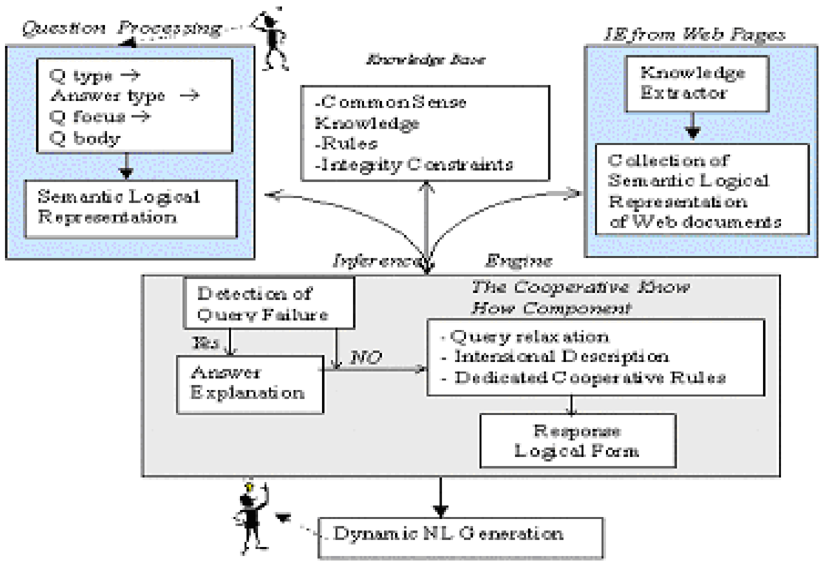


Figure 2.6: The WEBCOOP Architecture (Benamara, 2004)

However, the "cooperativity" only focuses on atomic and enumerative responses. The system utilizes a knowledge extractor and a robust question parser to select and examine the proposed answers. According to the cooperative rules, the WEBCOOP inference engine will determine the matching answers and organize them for output (Benamara & Dizier, 2003).

2.3.2 PiQASso

PiQASso (Antonio et al., 2001) is a QA system based on a combination of modern IR techniques and a series of semantic filters for selecting paragraphs containing a justifiable answer. Semantic filtering is based on several NLP tools, including a dependency-based parser, a POS tagger, a Name Entity (NE) tagger, and a lexical database. The flowchart of PiQASso is shown in Figure 2.7.

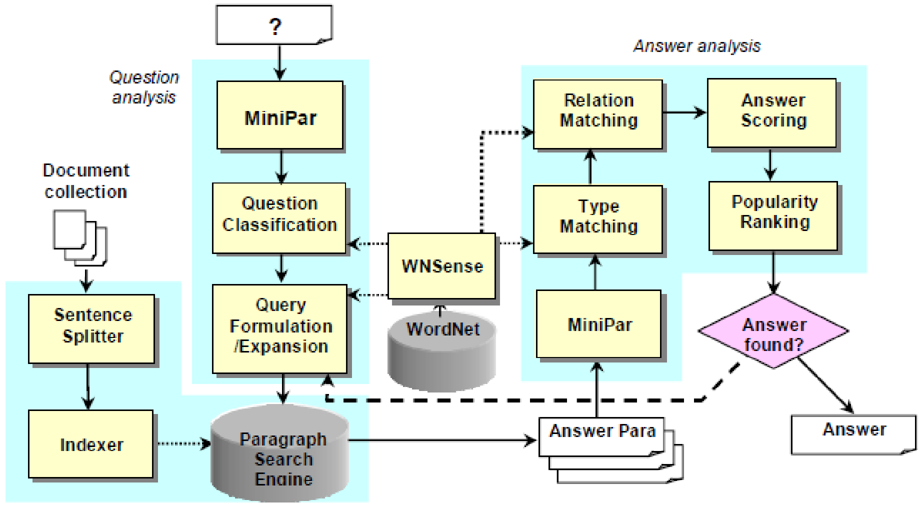


Figure 2.7: The flowchart of PiQASso (Antonio et al., 2001)

Semantic analysis of questions is performed in order to extract keywords used in retrieval queries and to detect the expected answer type. Semantic analysis of retrieved paragraphs includes checking the presence of entities of the expected answer type and extracting logical relations between words. Thereby, queries are expanded to cope with morphological variants of words by adding the synonyms of the search terms.

The answer retrieving in PiQASso is based on IXE (Attardi & Cistemino, 2001), a high-performance C++ class library for building full-text search engines. In PiQASso, sentences are parsed to produce a dependency tree which represents the dependency relations between words in the sentence. A dependency relationship is a binary relationship between a word called as head and another word called as modifier.

By checking entities and verifying that the answer sentence contains word nodes of the dependency tree that are of the same type and relationship with corresponding word nodes in the question dependency tree, matching words between question and answer are found. All matches between triples in the question and in the answer are considered. The greater the result of match distance the candidate sentence obtains, the more likelihood the candidate sentence is the most suitable answer.

2.3.3 AnswerBus

As an open-domain question answering system, AnswerBus (Zheng, 2002) is based on sentence-level web information retrieval. It accepts natural-language questions in multiple languages and retrieves relevant Web pages. From these Web pages, AnswerBus extracts sentences that are determined to contain answers. The working process of AnswerBus is shown in Figure 2.8.

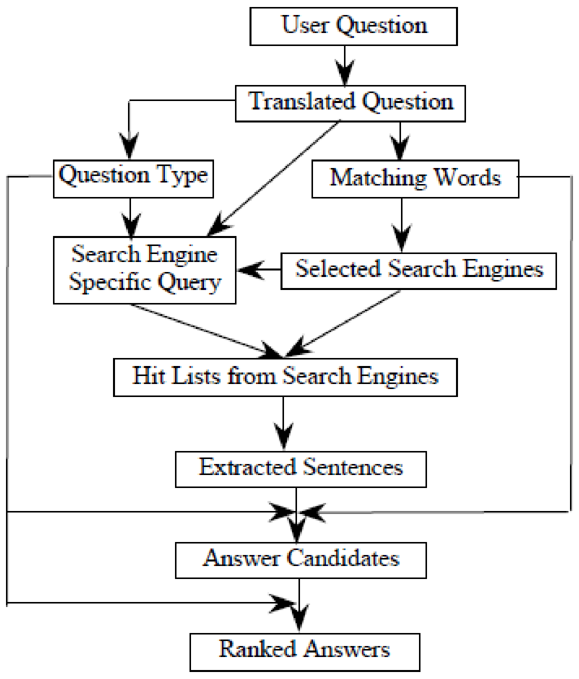


Figure 2.8: Working process of AnswerBus (Zheng, 2002)

The workings of AnswerBus comprise of mainly four steps:

1. selecting two or three search engines for information retrieval, and form search engine specific queries based on the question,
2. contacting the search engines and retrieve documents and retrieve documents at the top of the respective hit lists,
3. extracting sentences that potentially contain answers from the documents,
4. Ranking the answers and return the top choices with contextual link to user.

Instead of returning a snippet of fixed length of text, AnswerBus returns sentences, which can provide users with some contextual information of the answers. Meanwhile, the use of different search engines for different questions increases the likelihood that the answer sentences are among the retrieved documents. The result of this online QA system shows that it has higher accuracy than off-line QA systems.

2.3.4 AQUAREAS

AQUAREAS (Hwee et al., 2000) is a QA system based on machine learning approach, focusing on reading comprehension resources. The advantage of a machine learning approach is that it is more adaptable, robust, flexible, and maintainable. This approach comprised of two steps. First, a set of features are designed from question-sentence pairs to capture the information that helps to distinguish answer sentences from non-answer sentences. Afterwards, a learning algorithm is utilized to generate a classifier for each question type from the training examples. The example of the classifier is shown in Figure 2.9.

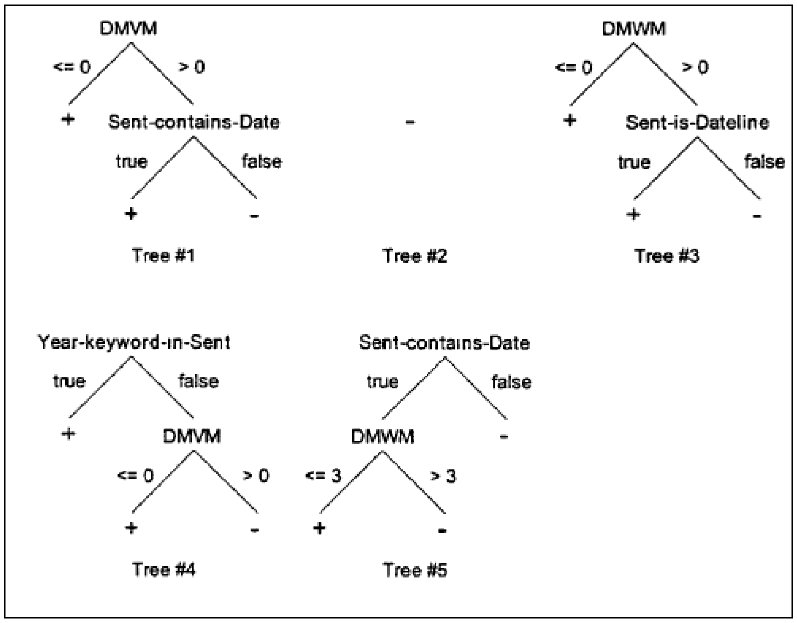


Figure 2.9: The classifier for WHEN question type (Hwee et al., 2000)

To each sentence, the classifier will decide if it is positive (an answer) or negative (not an answer) with a confidence value. The features that are considered in identifying the answer consist of named entity, co-reference information, keywords in questions. AQUAREAS also considers the same set of features in answer sentences. Compared to handcrafted rule-based methods, the machine learning approach avoids the need for continuous improvement or maintenance on the set of rules. Moreover, the results of this QA system are comparable to the results from other reading comprehension QA system.

2.3.5 START

START (B.Katz, 1997) is an information server built at the MIT Artificial Intelligence Laboratory. Since December 1993, START became the first natural language system available for question answering on the World Wide Web.

The START server is built on two foundations: the sentence-level NLP capability and the idea of natural language annotation for multi-media information segments. T-expression is employed to handle embedded sentences. Together, S-rules are implemented to help T-expressions which involve the verb expression and which meet the additional structural constraints. These two foundations are respectively implemented by two modules: (1) an understanding module while analyzes the English text and (2) a knowledge base which incorporates the information found in the text. Given an appropriate segment of the knowledge base, a generating algorithm produces English sentences. A user can retrieve the information stored in the knowledge base by querying it in English. The system will then produce an English response.

2.4 Discussion of QA

Besides the literature that was surveyed, there are many other techniques that can be exploited to improve the answer searching ability of QA systems according to their respective different domains. However, essentially, the ultimate goal of any answer extraction algorithms is to increase the accuracy of the retrieved answer candidates. For this purpose, researchers have utilized NLP tools, statistical methods and traditional IR techniques in the different phases the of QA system. Semantic techniques have also been employed to achieve optimal answers.

From the survey, WEBCOOP utilizes an inference-based method with the concept of cooperative answering with the aim of guiding computers to recognize accurate answers and to handle situations when no suitable answer is retrieved. The idea of introducing the cooperative method into QA systems is quite helpful for optimizing the QA search ability.

The strategy of PiQASso, however, is to integrate NLP tools (such as a Name Entity tagger) and modern IR techniques. As a result, the performance of PiQASso is very much dependent on its parser. In PiQASso, it also appears that the answer type identification and keywords are not very helpful for answer matching.

As the first online QA system, START utilized NLP annotation and sentence-level analysis. As a relative mature product, the response speed and answer retrieval accuracy of START is pretty optimistic. AQUAREAS retrieves answer by utilizing a machine learning approach, which decides on answer candidates by considering 20 features of questions and answer sentences in documents on the Remedia reading comprehension data set. Although the achieved accuracy from the machine learning approach is competitive compared to handcrafted algorithms, the performance from AQUAREAS is still not optimistic.

In contrast, AnswerBus, following the typical process of QA, exploits multiple IR techniques to retrieve the sentence level answers. As an open-domain QA system, the performance of AnswerBus is better when compared to PiQASso and AQUAREAS.

For rule-based QA systems, it was observed that the focus is on learning rules based on similar keyword detection or lexical relations between question and answers.

For instance, the manually derived heuristic rules of Quarc focus on lexical relations in different WH-type questions from a reading comprehension corpus. Quarc is a simple and efficient QA system employing a set of heuristic rules. However, it seems rather simple to handle other types of question-answering tasks.

In TextRoller, the generation of pattern rules is decided based on heuristics, using the training data from the TREC data set. The accuracy of TextRoller is greater than the accuracy of Quarc. The result of TextRoller shows that the indicative patterns work well for QA systems. However, the performance might be better if TextRoller could be developed with more appropriate tools for indicative pattern generation.

As mentioned earlier, the pattern rules of Webclopedia are learnt based on simple surface patterns. The rules of Webclopedia are categorized according to the question typology, which reveal the semantic relations between questions and answer sentences. However, the semantic relations are too complex to complete the decision trees of the rules. Also, lengthy question sentences seem difficult for Webclopedia to figure out.

However, researchers are also trying to generate rules using automatic or semi-automatic approaches to prevent humans from inadvertently missing out certain rules. To detect pattern relationships automatically for QA systems, there are different methods utilized in rule learning. In AnswerFinder, the patterns are learnt based on logical information instead of meaning. The graphs connecting between patterns