Learning Automatic Question Answering from Community Data

by

Di Wang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Although traditional search engines can retrieval thousands or millions of web links related to input keywords, users still need to manually locate answers to their information needs from multiple returned documents or initiate further searches. Question Answering (QA) is an effective paradigm to address this problem, which automatically finds one or more accurate and concise answers to natural language questions. Existing QA systems often rely on off-the-shelf Natural Language Processing (NLP) resources and tools that are not optimized for the QA task. Additionally, they tend to require hand-crafted rules to extract properties from input questions which, in turn, means that it would be time and manpower consuming to build comprehensive QA systems. In this thesis, we study the potentials of using the Community Question Answering (cQA) archives as a central building block of QA systems. To that end, this thesis proposes two cQA-based query expansion and structured query generation approaches, one employed in Text-based QA and the other in Ontology-based QA. In addition, based on above structured query generation method, an end-to-end open-domain Ontology-based QA is developed and evaluated on a standard factoid QA benchmark.

Acknowledgements

I would like to thank all the people who made this possible. I wish to express profound gratitude towards my supervisor Dr. Ming Li, for his invaluable support and discussions throughout my master study. My sincere thanks also go to Dr. Grant Weddell and Dr. Olga Vechtomova for reading my thesis. Lastly, many thanks go to my girl friend Yan Yan, my parents, and all my friends for their encourage and support.

Dedication

This is dedicated to my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since the invention of digital computers, electronically stored information has grown exponentially and has covered human knowledge in virtually all aspects. For instance, the World Wide Web (WWW), as a gigantic warehouse of information, meets a wide range of information needs. However, the explosive growth of WWW also increases the difficulty for users to efficiently acquire the demanded knowledge from scattered pieces of information. Although modern search engines can quickly provide user plenty of web links related to input keywords, users still often have to spent plenty of time on locating the precise information they are looking for. Clearly a new paradigm is wanted to allow more natural and more immediate access to their answers.

Meanwhile, making computers to understand human language gradually becomes possible with Natural Language Processing (NLP) technologies. Large collections of annotated corpus, user generated content, and structured knowledge are recently available to NLP researchers and massively enhanced the NLP abilities. Supported by the vast power of modern computers, NLP technologies now enable the Information Retrieval (IR) to shift its input from keywords to natural language questions, and shift its output from piles of related documents to concise answers.

1.1 Question Answering

The task of Question Answering (QA) is to find specific and concise answers (usually in form of phrases or sentences) to questions posed in natural language. For example, a QA system can accept natural language questions like "Who is the first president of United States?", and attempt to output the exact phrases or sentences expressing the answer "George Washington" based on its information sources such as a collection of web pages.

QA dispenses with unnecessary user interactions with typical search engines and web pages and is able to offer brief description of answer, which ideally serves the information needs better on interaction restricted environment such as Short Message Service (SMS). With regards to the input, it accepts natural language questions rather than query syntax, which makes it easier to use for the casual user and might better suit the voice-activated information retrieval tasks. The human language nature of its input question and answer sentence makes QA systems a effective choice to help achieve cross-language information retrieval as well.

Depending on different sources of knowledge, QA systems can be categorized into mixtures of following three types:

- Text-based QA systems, which retrieval answer text snippets from one or more free text documents (e.g. Internet web page).
- Ontology-based QA systems, which query answer entry from structured knowledge bases (e.g. expert domain knowledge).
- Community QA systems, which refer archived answer by linking input question with previous answered questions in datasets (e.g. Online Community QA site).

The next few chapters will introduce characteristics, components, and standard workflows of above three kinds of QA systems.

1.2 Question Analysis

The Question Analysis component is commonly the first step of all kinds QA systems. Intuitively, if question analysis module can not provide enough information from question,

the following components are less possible to discover correct answers to respond input question. Therefore end-to-end accuracy and recall of a QA system is very sensitive to the quality of its question analysis part.

The ultimate goal of question analysis is to interpret from plain text question sentence into the meaning of the user's inquiry. Although fully modeling semantic meaning of natural language sentences is one of most difficult problems in the field of artificial intelligence and also known as one of AI-Complete [62] problems, existing QA systems partially interpret user's intention by generating features from input question.

In Text-based QA systems, constructing keywords extension is commonly a must-have task in question analysis module. Given question "Who is the wife of Bill Clinton", if the IR module wishes to retrieval passages like "Hillary Rodham married Bill Clinton at their home in Fayetteville, Arksansas.", the question analysis component is responsible to feed IR module with query keyword extension *(wife OR married OR spouse) AND Bill Clinton*.

In Ontology-based QA systems, information is store in structured knowledge bases with certain scheme, and is only accessible by structured query corresponding to the schema. Therefore, the question analysis is primary in Ontology-based QA systems. It is responsible to parse and understand the natural language input and map it to structured query in format such as Sparql [59].

To produce richer feature set for Question Analysis, researchers in Question Answering community already introduced a wide range of Natural Language Processing (NLP) technologies to Question Analysis task, including Named Entity Recognition (NER)[49], syntactic and semantic parsing [23], coreference resolution [53], semantic role labeling [63]. Additionally, new features also were created specifically for better interpreting Question intention, such as UIUC Question classification [36].

1.3 Motivation

As mentioned in previous sections, the ability of understanding natural language question fundamentally differentiates Question Answering paradigm to keywords based search engine. However, there is still tremendous room for improvements in existing QA system to parse and model semantic structure of input question.

State-of-art QA systems often adopt off-the-shelf NLP tools and resources to generate bags of features without much tunning. Various features can be generated by these NLP tools to express the question such as pos-tagging, keywords query with extension, term weighting, semantic roles, named entities, and syntactic trees. Nevertheless, these NLP tools are built for general purpose of analyzing sentence structure and are not optimize for the QA task. For example, input sentence "where is MIT" should be extend to query pattern "locate in". Sometimes hand-crafted rules or manually labeled data are added to QA systems to further improve question analysis ability. However, it makes the QA systems expensive to construct and hard to export to other domain schemes or languages.

When human provide an answer to certain question in natural language form, the intelligent gap between the question and answer pair is worth exploring for automatic question answering task. Therefore, this thesis investigates, designs, and implements two applications to learn from community answered question-answer archive to understand the semantic intentions of natural language questions.

1.4 Contributions

The work of this thesis explores possible ways of making use of user-generated content in Community Question Answering (cQA) sites to implement automatic Question Answering functionalities, especially the question analysis component.

Specifically, this thesis proposes, implements and evaluates two novel approaches for automatic text-based and ontology-based QA tasks:

- A CRF-based passage extraction method is developed to explore the potential of the cQA based Question Analysis module for text-based QA. The major contributions are:

    1. Discovering a new query expansion method based on cQA archive;
    2. Designing a CRF-based passage extraction module to apply above query extensions as a feature;
    3. Showing how cQA-based query extension can improve the performance of passage extraction.

- A complete open-domain ontology-based QA system is implemented with a new cQA based training process for its question analysis module. The major contributions for this part are:

    1. Implementing a large scale question answering system to query in natural language to structured knowledge base;
    2. Inventing a new training method to learn mapping natural language questions to structured query based on community question answering pairs;
    3. A named entity recognizer with its dictionary is extracted from links in Wikipedia articles.;
    4. The normalization and scoring functions to rank and choose the best answer among various kinds of generated queries;
    5. Evaluation that shows this ontology-based QA system achieve high performance on both answer results and speed.

1.5 Thesis Organization

Chapter 2 introduces the background of this thesis including characters of QA system with different knowledge source, state-of-art question answering systems, evaluation methodology for question answering, and related question answering components and resources.

Chapter 3 discuss the basic observation and assumption in this thesis, and the following thesis following this aspect to conduct research.

Chapter 4 deploys cQA resource to text-based QA task. It consider the problem of extending the query keywords set for passage retrieval.

Chapter 5 describes full processes of a new ontology-based QA system, particularly the training from cQA data and scoring methods to rank best answer.

Finally, chapter 6 concludes the thesis, and discusses future directions for applying cQA resource for automatic QA task.

Chapter 2

Background and Related Work

This chapter wishes to give the reader a grounding in the state of the art of Question Answering (QA) research, and to identify issues that are addressed by this thesis.

2.1 Text-based Question Answering

Unstructured question answering systems normally integrate natural language processing (NLP) with information retrieval (IR) to achieve QA functionalities. As shown in Figure 2.1, those QA systems employ an abstract pipeline architecture consisting of three components: question analysis, information retrieval (document retrieval and passage retrieval), and answer extraction. First, question analysis module is responsible for parsing the input question, determine the type of question, and generate query keywords with extension and constraints. Secondly, information retrieval module search documents and that contains query keywords and matches other constraints. Finally, the answer extraction module analysis of the content returned from IR module then selects the most likely word, sentence or passage to answer the question.

When a QA system accepts a natural language question, question analysis module needs to parse this question and extract meaningful structured information (parameters) from the input question for later modules. Question analysis module may parse grammatical structure such as part-of-speech tagging, phrase structure trees, or grammatical relations

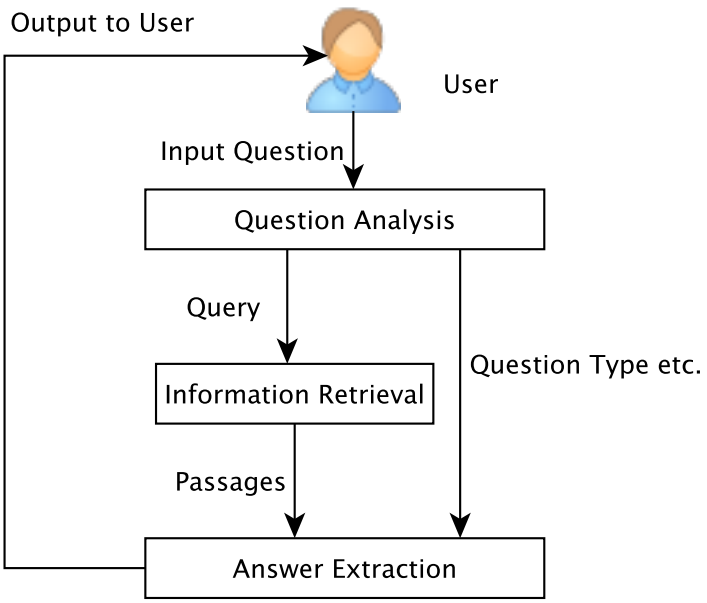

Figure 2.1: A Basic Text-based Question Answering Architecture

(typed dependency) formats. In addition to grammatical parsing, question analysis module could also be required to determine the type of the question and then the type of answer. For example, it could classify "how many" as numeric values question class such as the classifier developed by Li and Roth [37]. Finally, question analysis module should extract keywords and/or generate keywords' extensions or other features from the input question. Its purpose is to construct an effective query to retrieve documents or passages containing related features as candidate answers. For instance, some modern QA systems may formulate keywords with certain query syntax for its IR components, such as *president AND (United States OR America)*.

In open-domain question answering systems, based on the analysis [41] from Jimmy Lin about the impact of the answer context, retrieving relevant paragraphs may have better overall user experiences than both whole documents and word phrases. Therefore, this thesis only considers passages (groups of sentences) as finer grained answers than document. Passage retrieval is one of retrieval methods that aim to search compact text

excerpts within documents in response to users' queries. It generally involves extracting proper passages from documents, and further ranking passages based on their relevance to IR query. Given a query, it can either rank documents first then extract relevant passages, or extract relevant passages from all documents then rank those passages.

2.1.1 Query Expansion

To help increase the recall rate of relevant documents retrieved by IR module, many kinds of query expansion or constrain methods have been introduced to the QA field, which is vital for those questions with few available extracted keywords to feed to IR module.

The most straightforward would be expansion using morphology derivations or synonymy from WordNet [51]. Hovy et al. [24] and Greenwood [19] have shown improvements of IR quality with restricted use of WordNet resources. Pasca and Sanda [55] manually maintained a rule-based method to inserting morphology and lexical derivations of original keyword. Lin and Pantel [39] present an approach in which questions are syntactically paraphrased to more flexible answer matching. Similarly, Hermjakob et al. [22] reformulate the questions with semantic paraphrases, called phrasal synonyms, to enhance the retrieval performance. Negri and Kouylekov studied how to account for syntactic and semantic correspondences between questions and passages, by computing similarities between their respective syntactic trees [54] and an edit distance. Therefore, those query expansion methods basically attempt to extend the original query by different expressions with similar meaning in given context.

Beyond keyword similarity, several researchers have tried pre-processing the corpus and matching question focus (e.g. "where") to structurally annotated entities such as Lin et al. [40], Bilotti [3], and the predicate annotation by Prager et al. [58]. In addition, Riezler et al [60] do query expansions by translation based methods. Derczynski et al [16] has tried identifying the difficulty of input questions and then inserting extra search keywords for difficult question, which discovered extensions from previous QA evaluations' answer text.

2.1.2 Passage Retrieval

Passage retrieval is a method that matches highly relevant paragraphs to the input inquiry. Because it is able to discover those potential answers surrounded by relevant passages but in irrelevant documents, the passage retrieval is widely used in text-based QA systems.

The concept of passage retrieval was first proposed by Salton [61] in the 1993 which suggested applying certain strategy to identify that two text fragments is similar to each other. Passage retrieval algorithm do match and sort passages related to inquiry. Some classical passage retrieval algorithms retrieval and rank passage according to keywords frequencies, such as MITRE [38]; Keywords densities are also take into consideration, such as in MultiText [13], Site Q[34], and IBM [25]; Moreover, language model based algorithm [42] to retrieval and rank passage. There is also many method in the field that tried to retrieval passage with probability techniques. In [52], Mittendorf and Schanble considered passage retrieval as stochastic process, which generates text fragments independent of any particular query. Hidden Markov Models (HMMs) was used to model this stochastic processes. In [27], Jiang and Zhai thought most existing work tends to rely on pre-segmenting documents into fixed-length passages which are unlikely optimal because the length of a relevant passage is presumably highly sensitive to both the query and document. They handled this problem using HMM-based method, which naturally the topical boundaries between sentences that irrelevant to its neighbors. Melucci [50] presented a Bayesian framework, the probabilistic technique, to implement passage retrieval from texts having a large size or heterogeneous semantic content. The proposed technique is independent on any supporting auxiliary data, such as text structure, topic organization, or pre-defined text segments.

The passages themselves can vary in size and degree of overlap. A critical problem in passage retrieval is to accurately locate the boundaries of coherent relevant passages in a document, which is called passage extraction. The performance of passage retrieval relies on accurate passage extraction [27]. In [8], the authors classified the existing approaches of passage extraction into three categories: text block relativity [29], text segmentation [9] and probability models [42]. Despite its importance, the passage extraction problem has not been seriously addressed in existing work. None of the existing passage retrieval methods was intentionally designed to achieve the goal of extracting passages that are both query-dependent and coherent[27]. It is trickier to consider the dependency that exists be-

tween chunks of texts (in this project, sentences)[8]. Khalid [31] compared the effectiveness of several passage retrieval techniques with respect to their usefulness for QA. Their main interest was the contribution of sliding passages as apposed to disjoint passages, and evaluated the retrieval approaches on two different QA tasks: factoid-QA and a relatively new problem in the QA field: that of answering why-questions (why-QA). In [17], the authors presented a graph-based model to represent the sentence dependencies. That model is a generalisation of the Hits algorithm. The similarity function considers the structural similarities. [66] concluded that neglecting crucial relations between words is a major source of false positives for current lexical matching based retrieval techniques. The reason is that many irrelevant passages share the same question terms with correct ones, but the relations between these terms are different from those in the question. To address this problem, [14] proposed a novel fuzzy relation matching method which examines grammatical dependency relations between question terms to improve current passage retrieval techniques for question answering. The authors employed Minipar to accomplish dependency parsing, and presented a statistical technique for measuring the degree of match of pertinent relations in candidate sentences with their corresponding relations in the question.

2.2 Ontology-based Question Answering

The rise of Semantic Web has provided a platform for users to generate and share knowledge. In recent years, large structured knowledge bases, such as those in Linked Open Data (LOD) [4], already contain billions of RDF (Resource Description Framework) [47] triples and act as a rich source of information for diverse user needs. To bring the advantages of these valuable data to casual users and further answer their questions, natural language interfaces to access those knowledge databases are highly demanded.

Ontology-based Question Answering systems utilize knowledge stored in structured data store. It also can been viewed as natural language interface to structured databases. Unlike text-based QA which is typically agnostic to the text content of its data source, the knowledge source of early structured QA systems were restricted to closed domains (e.g. a geography database). However, with the recent emerge of Linked Data concept, DBpedia data set [5], and commercial interest such as Google Rich Snippets[1], huge amount of

[1]https://developers.google.com/webmasters/richsnippets/

heterogeneous structured knowledge (metadata) is now available and opens new challenge to utilize these structured knowledge to QA field.

The major task of structured QA is to generate proper structured database query based on natural language input. With respect to input, many structured QA systems require Controlled natural languages (CNLs) as input to enable accurate mapping from question to predefined terminology and schema of data source. But easy to use and flexible natural language input without restriction is obviously desired by casual users. Hence the Question Analysis component of a ontology-based QA system significantly affects its usability. In many large open-domain ontology-based QA systems, the problems of knowledge sparseness, duplicity and incompleteness might exist in its knowledge base. For that reason, answer selection is also needed to choose and validate the results returned from multiple queries.

In fact, the research on accessing structured database with natural language was started more than forty years. Since 1970s, William Woods built one of the world's first natural language question answering systems (LUNAR) [72] for two databases about chemical analyzes and the literature references of moon rocks. The translation from question in English to its database query language was done by a rule-driven ATN parser [71]. Soon after, several other systems with larger databases or larger vocabulary were developed such as LADDER [21] and TEAM [48]. Zelle and Mooney [74, 75] investigated learning approaches, specifically learning to parse learning to parse into Prolog queries. These work of Mooney focused on applying Inductive Logic Programming to generate semantic grammar from examples in certain domain. Popescu et al. invented PRECISE [57, 56] which guaranteed the correctness of its output by identifying natural language queries that it can not fully understand.

Recently, the field of natural language interface to structured data focused on ontology knowledge base, with large number of newly available ontology data. The task becomes translating a natural language query into structured query language for retrieval answer knowledge statements. Currently, the most popular query language is Sparql [59] for RDF. Moreover, several question answering systems recently developed based on ontology knowledge base. ORAKEL [11, 12] system introduced by Cimiano et al. rigorously parses input using lambda calculus. AquaLog [44] and its successor PowerAqua [45] map question into sets of triple text segments, matches to most similar ontology entities, and then generate triple queries to find answer. Similar with PANTO [69] and Querix [30] are implemented

with similar approaches as AquaLog, and cover geography, restaurants and jobs domains. Damljanovic et al. desinged FREyA [15] system which emphasize user interaction and feedback. However, these ontology-based QA systems are both close-domain and strongly reply on vocabularies derived from ontology nodes and WordNet extension.

2.3 Community Question Answering

Community Question Answering (cQA) makes use of human intelligence to resolve questions. The major responsibilities of a cQA engine are retrieval archived answers for new question and find different potential answerer for different question.

Based on the observation that same question might be repeatedly asked by different users, preparing a list of question with answers (FAQ) has been a convenient way to share knowledge since the early days of WWW. Recent emergenced Community-base Question-answering services such as Yahoo! Answers [2] and the WikiAnswers [3] already accumulated large volume user generated knowledge. For example, Yahoo! Answers alreay achieved 300 million questions [4] and more than one billion posted answers [5]. Figure 2.2 shows a example question and its best answer discussed on Yahoo! Answers.

Unlike text-based QA and ontology-based QA, the resources in cQA sites directly serve the information need in natural language question form. Hence, when users query full sentence queries which beyond the paradigm of simple keyword-document relevance, modern search engines resort to display previous answered question from cQA site to help user efficiently find answer. Si et al. [64] reported that 25% of Google's search result first page contain at least one link to Q&A sites in China.

Most research carried on cQA field mainly focus on effectively retrieval answers from cQA archive, question intention analysis, and social interactions on cQA. Xue et al. [73] and Jeon et al. [26] investigated the combination of translation-based model to compare question similarity and query retrieval model to evaluate answer. Wang et al. [70] reported

[2] http://answers.yahoo.com
[3] http://wiki.answers.com
[4] http://yanswersblog.com/index.php/archives/2012/07/10/we've-reached-300-million-questions/
[5] http://yanswersblog.com/index.php/archives/2010/05/03/1-billion-answers-served

Figure 2.2: An Example Yahoo! Answers page shows a question with its best answer.

that the syntactic tree based question comparison achieves higher performance than bag-of-word or tree kernel based. Cao et al. [7] suggested the category information can improve cQA retrieval performance. Li et al.[35] and Bian et al. [2] tried automatically identifying whether a question is personal or subjective. Liu et al. [43] and Jurczyk and Agichtein[28] investigated ways of finding experts from users' behavior in the community, which enable forwarding unanswered question to certain experts in the community.

2.4 Evaluation

Text REtrieval Conferences[6] (TRECs), organized by the U.S. National Institute of Standards and Technology (NIST), has organized large-scale evaluations for QA systems annually between 1999 and 2007. Each year, the QA task involves a new set of questions

[6]http://trec.nist.gov

and answer nuggets that can be found in corpuses likes AQUAINT which contains approximately 1 million articles in 3 gigabytes of text. Documents in those corpuses covering archives such as Financial Times Limited, Congressional Record of the 103rd Congress, the Federal Register, Associated Press Worldstream News, New York Times News, and the English portion of the Xinhua News. All those answers submitted to the evaluation and judged correct by evaluators can be also used to train and evaluate other QA systems. Similar with TRECs, the Cross-Language Evaluation Forum[7] (CLEF) in Europe organized evaluations in monolingual and cross-lingual QA since 2003. The Mooney GeoQuery dataset [75] contains 250 questions and their structured logical expressions about United States geography information. This dataset is extensively used to both train and evaluate the natural language interfaces to database, such as PRECISE.

[7] http://www.clef-campaign.org/

Chapter 3

Observations and Assumptions

3.1 Question Topics and Focus

There are many possible perspectives to understand one natural language question. When it comes to Information Retrieval, one of our duties is to find the most related objects from large amounts of objects. It is naturally to see that there are questions that are easier to be solved with the help of IR system, which have strong *topic* among the question. For example, questions like "Who stared the movie Avatar?". We could easily locate those pages contain the answer for it by searching only keywords "Avatar". While, a counter example could be a question likes "What is the sum of 231 and 213?". It is not likely that the second kind of question could be answered by searching existing text in a corpus without deep semantic understanding. Therefore, in this project, we consider those questions always contain "topic". Here, the "topic" is the major object in a question that represents the users' intention.

Several reasonable ways exist to discover the topic part from input question. We could apply name entity recognition parser to detect name like word from the sentence. Using grammatical parser like Stanford parser [32] to get proper noun as topic. Also, term frequency also could employ as judgment for topic selection. It means that if some word in the sentence has lower term frequency than others. This term tends to be in topic words. On the other hand, the topic could also be user marked (human selected) as it did in the TREC QA track from 2004 to 2007.

After selecting the topic part of the question, we could get remaining parts as the question focus. For example, in question 'Who wrote the book Hamlet?", Hamlet is the topic and the else part can consider as the question focus. (Perhaps, we could continue to divide the non-topic parts if the question is rather complex.)

As discussed in chapter 2, the passage retrieval modules is to locate the most likely passage as the query described [6] . And, if we define the target passage T of our QA system have the form $X_{front}AX_{behind}$, where the A is the answer and X is the text surrounded it. Then, the question answering problem could be considered as to generate the query that matches X as much as possible. In this thesis, we attempt to design schemes and do experiments to produce queries for matching the sounding text X.

Now we could consider topic part and focus parts separately when doing query expansion and retrieval. Intuitively, topic and focus parts act different when answering a question. The topic part used to be more "fixed" or "discriminative" than focus parts and highly likely should appear near the answer. While the focus part plays another role on answers' surrounding passage X, it may not appear in X but there should at least some terms corresponding to focus part shown in X. Formally, we could simply define the target passage T is a set of words contains {topic part, f^*(focus), answer}, where f^* is a mapping from focus part of the question to the corresponding words should near answer.

3.2 Three Classes of Question-Query Transform Functions

Question answering is such a subset of IR problems that users want to retrieve the information that he doesn't know. This indicates that the input question doesn't likely contains enough direct searching keywords for answer target. Question-query transformation could be employed as an important method to overcome this gap. With the topic and focus perspective introduced above, we could therefore define three kinds of question-query transform function at different granularity.

First, equation class function, is to find the extent words have the same sense of meaning on the condition of the original question or only topic. This could be done by validating each question word's possible synonyms by fitting the original question or by checking high

co-occurrence with topic words in corpus. Or as works did in [60] that using translation based method to exam that whether the some replaced new word doesn't change the meaning (translation) of original question. For example, 'Who starred the movie Avatar?" equals to 'Who is the leading actor of the movie Avatar?".

Second, correspondence class function, is to find the corresponding keywords or sentence pattern in answer's context with the *focus* part of the input question. For example, the "Who wrote the book *?" should correspond to keywords like "author". This may be done by off-line statistic or machine learning on the focus parts of the question in the question answering community archive. This class of transformation is one major feature in Chapter 4.

Third, inference class function, is to find the meaning or concept highly close to the topic parts, like abbreviation or an equivalent concept like "44th President of US $\rightarrow$ Barack Obama".

After we obtain the query from the transformation function of the input question, our attention moves to design passage extraction model to combine and match those rich features of input question.

Chapter 4

Extending Text-based QA with Community QA

In this chapter, we preset a Conditional Random Fields (CRF) based model to implement passage extraction.

4.1 Challenges

Let's consider one example first:

TREC 2007 QA track data
Question Number: 239.2
Question: Who was the first host of Jeopardy?
Answer: alex trebek;Art Fleming
Example Text:

Jeopardy! is an American quiz show featuring trivia in history, literature, the arts, pop culture, science, sports, geography, and more. There are also wordplay categories. The show has a unique answer-and-question format in which contestants are presented with clues in the form of answers, and must phrase their responses in question form. **The show has**

a decades-long broadcast history in the United States since its creation by Merv Griffin in 1964. It first ran in the daytime on NBC from March 30, 1964 until January 3, 1975; concurrently ran in a weekly syndicated version from September 9, 1974 to September 5, 1975; and later ran in a revival from October 2, 1978 to March 2, 1979. All of these versions were hosted by Art Fleming. *Its most successful incarnation is the Alex Trebek-hosted syndicated version, which has aired continuously since September 10, 1984, and has been adapted internationally.*

The passages we look for can be of various lengths, which depend on different documents. In this example, the correct answer is Art Fleming. According to our human-being reasoning, we have to infer the answer from the first sentence to the third sentence in the underlined part to get the right answer. We hope our model and algorithm can handle this kind of reasoning partly. How to develop a machine learning method based on a training corpus of documents, which can take full advantage of the inter-sentence relationship and rich features? In this project, we present a new approach trying to tackle this passage extraction problem.

The example above motivate us about how human being or a computer extract a passage from a random document by posing this problem as a sequence labeling problem. Intuitively, a document can be regarded as a sequence of sentences that can be partitioned into several segments where each segment is relatively coherent in content and every sentence is dependent on its neighbors logically. As a human being, we have to infer the correct answer from several consecutive sentences in a segment by reasoning and extract the segment as the retrieved passage. The sentences in the document contain some information , which is called features in CRF, such as number of key words, length of the sentence, similarity between it and the question and whether it is related to the question type, etc. So, we need to read the document from the beginning to the end and extract informative passages consisting of neighboring sentences. This procedure is a kind of sequence labeling, a label of 1 denoting the extracted sentence sentences and 0 denoting the non-extracted sentences.

4.1.1 CRFs Model

From the aspect of machine learning, we consider passage extraction as a sequence labeling problem. In our model, each document is considered as a sequence of sentences and the purpose of passage extraction is to label the sentences in the sequence with 1 and 0, where a label of 1 indicates that the sentence is a extracted sentence for the passage while 0 denotes a non-extracted sentence. The label of one sentence is expected to be involved in the labels of other neighboring sentences.

Conditional random fields (CRFs) [33] can helps us to achieve this objective. CRFs is simply a conditional distribution $p(Y|X)$ with an associated graphical structure. Because the model is conditional, dependencies between the input variables X do not need to be explicitly represented, affording the use of rich, global features as the input. A CRF framework is constructed by adding all possible features which may be complex and overlapping. Further, a set of features for the task of passage extraction is introduced to the CRF model. As a discriminative model, this sentence based CRF model outputs the maximum likelihood of the extracted sentence sequence, conditioning on the whole sentence sequence which is the document.

In this paper, we only focus on the linear-chain CRFs, which can capture the dependency between contiguous sentences. Given a question Q, and a retrieved document X that is relevant to Q. For the simplicity, we represent X as a sequence of sentences, that is, $X = (x_1, x_2, x_3, \ldots, x_T)$. For the given question Q, our goal is to find the corresponding labeling sequence of D, $Y = (y_1, y_2, y_3, \ldots, y_T)$. The linear-chain conditional random field is a distribution $p(Y|X)$ that takes the form.

$$p(Y|X) = \frac{1}{Z(x)} exp\{\Sigma_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, X)\},$$

where $Z(x) = \Sigma_y exp\{\Sigma_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, X)\}$ is a normalization function; $\Lambda = \{\lambda_k\}$ is parameter vector; $\{f_k(y, y', X)\}_{k=1}^{K}$is a set of binary feature functions over the entire observation sequence, which can describe any aspect of transition from y_{t1} to y_t and the global characteristics of X.

Parameter Estimation Given the training data $\mathbb{D} = \left\{(X^i, Y^i)_{i=1}^{N}\right\}$, where each $X^i = (x_1^i, x_2^i, \ldots, x_T^i)$ isa sequence of the desired extractions for the document. Parameter estimation of Λ is typically performed by maximum log likelihood, $ml(\Lambda) = \sum_{i=1}^{N} \log p\,(Y^i|X^i)$.

Perhaps the simplest approach to optimize $ml(\Lambda)$ is steepest ascent, but this requires too many iterations to be practical. Newtons method converges much faster because it takes into account the curvature of the likelihood, but it requires computing the Hessian, the matrix of all second derivatives. Particularly successful have been quasi-Newton methods such as BFGS , which compute an approximation to the Hessian from only the first derivative of the objective function. As an alternative to limited-memory BFGS, conjugate gradient is another optimization technique that also makes approximate use of second-order information and has been used successfully with CRFs[65]. In our experiment, we use steepest ascent method.

Inference During training, computing the gradient requires marginal distributions for each edge $p(y_t, y_{t1}|X)$ and computing the likelihood requires $Z(X)$. We compute the most likely (Viterbi) labeling $Y^* = \arg_Y p(Y|X)$. In our linear-chain CRFs, both inference tasks can be performed efficiently and exactly by variants of the standard dynamic programming algorithms for HMMs. We define the forward variables $\alpha_i(y|X)$by setting $\alpha_i(y|X)$ equal to the probability of starting with state y and computer $\alpha_i(y|X)$ by the recursion $\alpha_{i+1}(y|X) = \sum_{y'} \alpha_i(y|X) \exp(\Lambda_i(y', y, X))$, where $\exp(\Lambda_i(y', y, X)) = \Sigma_k \lambda_k f_k(y_t = y', y_{t1} = y, X)$. The backward variable $\beta_i(y|X)$ recursion is exactly the same. Finally, we computer the globally most probable assignment $Y^* = \arg\max_Y p(Y|X)$ by using the Viterbi recursion[65].

4.1.2 Features in CRF models

The main features in our linear-chain CRF models for passage extraction are listed below.

Query Based Features:

- Length of the sentence
- Number of (topic, focus, whole, F_e, F_c,F_i) keywords in the sentence
- Density of (topic, focus, whole, F_e, F_c,F_i) keywords in the sentence

, where F_e, F_c, and F_i are three transform function "equals", "correspondence", and "inference" discussed in chapter 3.

The procedure to generate F_c is described as following:

1. Given input question Q^*, we divide it into $topic^*$ and $focus^*$.
2. Collect a supporting set of (Q, A) pairs from QA community such that for every $Q_i \in (Q, A)$ its $focus_i$ equals (or close similar) with $focus^*$.
3. Counting the terms in $A_i - topic_i$ for all selected supporting QA pairs
4. Select the term t, such that (frequency(t, $A - topic$) - frequency(t, background corpus)) larger than pre-defined threshold.

Coherent Features:

- Wether the sentence contain keywords in group i and its neighbor contains keyword in group j (for example, whether current sentence have keyword in topic and its neighbor contains keywords in group F_c).

Density of keywords in sentence is a classical way to evaluate if the sentence is relevant to the question. It can be calculated by the ratio of frequency of keywords in the sentence to the frequency of keywords in the collection. All the Query and Coherent based features are used to capture the dependency between contiguous sentences for CRFs. For example, if the density or the coherent of one sentence is great than given threshold, we consider this sentence should be extracted for the passage without hesitation. Moreover, if the neighour sentence contain one keyword, we would think this sentence can be included in the passage as further context. In addition, the feature function $f_k(y, y', X)$ can not only describe aspects of transition from y_{t-1} to y_t but also capture the global characteristics of X

4.2 Implementation

Viterbi Search for CRF

Dynamic programming Viterbi algorithm is introduced to solve CRF models. The running-time is linear with the number of sentences and quadratic with number of features, $\Theta(\#sentence \times \#feature^2)$. One observation of our practical problem is that even the feature's value is

unbounded in general, but its values usually small (0, 1, 2 or 3). That infers that we could hold a kernel matrix for computed results, because there are only polynomial times n possible kernels for small value n. We could compute each feature combinations for once and save computation time for the rest of them which just repeat the previous combination. Further, we even could compute these kernel matrix off-line, which makes running-time reduce to Θ(#sentence).

Data indexing and keyword-sentence-positions retrieval

Index is built based on the open source information retrieval software library, Apache Lucene[20]. For this specific project, besides recording document numbers and keywords frequency into posting lists, we also need to record detailed sentence number and positions of each keyword appear in that documentation. Hence, Lucene's index structure is modified to store these extra information with hopefully minimum changes. To finish sentence aware indexing, we also need to break the input text into sentences and gracefully handle cases like 'Mr.". Then Stanford parser's[32] sub-package is used to detect the end of sentences. Further, additional boolean search is implemented to support modified posting list (sentence aware, position stored) from stretch.

Passage Extraction

SAX model is used to parse TREC QA track document data sets AQUAINT and question sets. Then, retrieval Yahoo! Answer's QA set as described above as supporting QA pair. Once keywords are extracted from question, other features needed could directly be fetched from above index built above.

Dynamic programming Viterbi algorithm for linear CRFs model then labels sentences as sequence data, those marked 1 is selected as passages.

4.3 Experiments

In this section, experiments are conducted to test this CRF based passage extraction approach empirically. The data set is from TREC, which contains questions and judgment

in TREC QA task in 2004. The training is performed on TREC QA task 2000 questions and judgments that arbitrarily mark seven sentence long passages which center is answer keyword in answer documents.

First, the following is samples from focus community generated works for F_c. Here, for each focus part in a question, at most 500 QA pairs considered.

Topic : Tale of Genji
Focus : Who wrote it?
F_c (Who wrote it) : { wrote states writing long people version love author story time life words written write read man find music books job called book word years part }

Topic : The Clash
Focus : What kind of music does the band play?
F_c (What kind of music does the band play) : {bands pop rock hear time band friends metal work piano favorite country stuff start sound years hope song songs punk kind playing life money listen hard find sounds music feel jazz play instrument school guitar }

We could see that the most generated words list is related to each focu part in question. It seems highly possible these words would also appear in the target passage, which could improve the overall effectiveness. However, there are still some words looks not quite questionable dependent. This situation should be improved if more strict thresholds are settled.

Next, experiments are done on recall rates of CRFs passage extraction algorithm:

Transform	#ans sent	#CRF passage covered	total answer #doc
Community	1236	1033	1070
None	331	224	1070

In the above table "#ans sent" indicates the number of sentences containing answer in retrieved documents. "#CRF passage covered" means the sentence number in "#ans sent" that is covered in CRF extracted passages. And the "total answer #doc" is the total number of documents that contain answers.

Here, we notice that the CRFs passage retrieval method could both give good recall rate about $\frac{1033}{1236} = 0.84$ and $\frac{224}{331} = 0.68$.

And, we further could discover that the community-based could significantly increase the document retrieval recall rate and mild increase on passage extraction recalls rate with reasonable size of expansion.

Chapter 5

An Ontology-based QA System built from Community QA

This chapter introduces an approach for answering open-domain natural language questions over a structured knowledge base by translating the natural language questions into structured queries, such as database queries.

5.1 Problem Formulation

A simplified knowledge base can be expressed by a collection of triples (s, p, o), where each s refers to the subject; p indicates the predicate; and o is the object. Each subject must be a concept. The concept may indicates a real-world entity such as "Bob Dylan", or an abstract notion such as "Artist". Each predicate is a special concept representing the relationship between subject and object. And the object in a triple can be either a concept or a literal value. Then a knowledge base can be considered as a directed graph with labeled edges, where subjects and objects are nodes and predicates are edges. The figure 5.1 shows a example sub-graph describing "Bob Dylan" in a knowledge base. Additionally, a *path* in the knowledge graph is a sequence of predicate-labeled directed edges. In above example sub-graph, the path represents "the birth place of the artist that played the song" is $(musicalArtist, birthPlace)$. The inverse relation of p is denoted by p^{-1}, and (s, p, o)

is equivalent to (o, p^{-1}, s). For instance, one of the predicate paths from "Bob Dylan" to "Derek Boogaard" is $(birthPlace, deathPlace^{-1})$.

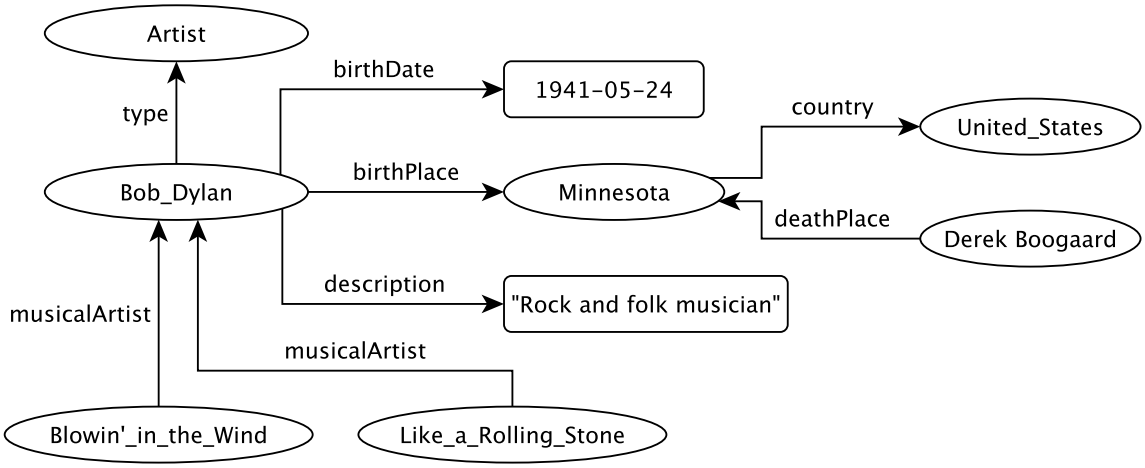

Figure 5.1: A example knowledge subgraph describing "Bob Dylan"

In knowledge bases, a structured query is consisting of a set of triple patterns known as a Basic Graph Pattern (BGP). Each triple pattern is similar to triples (s, p, o) except that each subject, predicate and object can be a variable. For example, a very basic BGP is finding o for given corresponding s and p, such as a given subject "Bill Clinton" and the predicate "child" to find variable "?children" that will be "Chelsea Clinton".

As input, we are given a natural language question, comprising a sequence of words $(w_1, \ldots, w_n)$. In the question, an entity surface text is a subsequence of words $(w_i, \ldots, w_{i+j})$ that can possibly refer to one or more entities which are subjects or objects in the knowledge base.

Our goal is to map the plain text question to a structured query. BGP of the structured query corresponds to all logical constraints in question. And each triple pattern contains proper mapping from an entity surface to an entity, and mapping from the question focus to a predicate. After mapping all logical constraints in the question into BGP, the structured query is expected to return entities or literals as few as possible that contain the answer to the question.

5.2 Challenges

In general, there are three challenges to answer natural language question with structured knowledge base.

- The first is recognizing entities in an unstructured question and associating those named entities with their corresponding entries in the knowledge base. The word or phrase in the unstructured question often ambiguously implies various database entries. For example, the word "Obama" can refer to both "Barack Obama (44th President of the United States) " and "Obama, Fukui (a city located in Fukui Prefecture, Japan)".
- The second is the mapping from each question's semantic intention of entities to database relations (the predicate in the triple) and then construct structured queries. For instance, the question "who is the wife of Bill Clinton" might link to the "spouse" relation to the entity "Bill Clinton" in the certain knowledge base.
- The third is identifying structured dependencies among database relations such as nesting and Boolean algebra.

Prior art solutions exist that attempt to address the above mentioned challenges. Generally speaking these prior art solutions either (1) manually matching whole input questions to structure queries, or (2) linking keywords in questions to names of entries in the database. The approach (1) requires a large number of hand-written mappings from questions to structured queries, which typically has a very low recall rate. Regarding (2), this type of approach often has low precision because of its nature of approximate matching between keywords and entities. Both types of approaches provide poor performance on large open-domain knowledge bases. Furthermore, above approaches involve maintaining substantive human-annotated mappings or domain-specific keywords, which makes them very expensive to extend and improve from the implementation perspective.

These challenges are a practical obstacle to design and implementation of Question Answering (QA) systems with structured knowledge that are accurate enough for widespread user adoption. There is a need for a computer system, computer program, and computer implemented method that addresses the above mentioned obstacles. Accordingly, what is

needed in the art is a new knowledge base QA system and a method that automatically translate natural language questions into structured queries that lead to the correct answers and require neither hand-written mapping patterns nor domain-specific approximate matching tuning for scalability.

5.3 System Architecture

The following presents a simplified summary of the Ontology-base Question Answering system in order to provide a basic understanding of some aspects of the innovation. This summary is not an extensive overview of the innovation. It is not intended to identify key/critical elements of the innovation or to delineate the scope of the innovation. Its sole purpose is to present some concepts of the innovation in a simplified form as a prelude to the more detailed description that is presented later.

In the view of forging situation, the present invention provides system and method to answer open-domain natural language questions with structured knowledge base by translating them to structured queries such as database queries.

In one aspect of the invention, a computer implemented Ontology-base QA method is provided comprising the steps of:

1. recognizing one or more Named Entities (database entries) in a natural language question;

2. retrieving pre-trained possible predicate (relationship) mappings using rest of question exclude Named Entities;

3. constructing structured queries in accordance with question's logical representation including Boolean algebra on basic graph pattern (BGP) and nested queries;

4. sorting possible structured queries with a score combing the confidence of Named Entities Recognition (NER), the confidence of BGP translation, and the confidence of queries nesting;

5. building natural language answers with one or more queried outputs.

Another aspect of the invention is to provide a training system and method that automatically learn the transformation rules, patterns, and statistics for translating natural language question into structured queries. The present invention integrates the knowledge base with training pairs of questions and answers, which discovers and links textual patterns with semantic relationships.

Figure 5.2 depicts the workflow of unsupervised training of the mapping from natural language questions to structured queries.

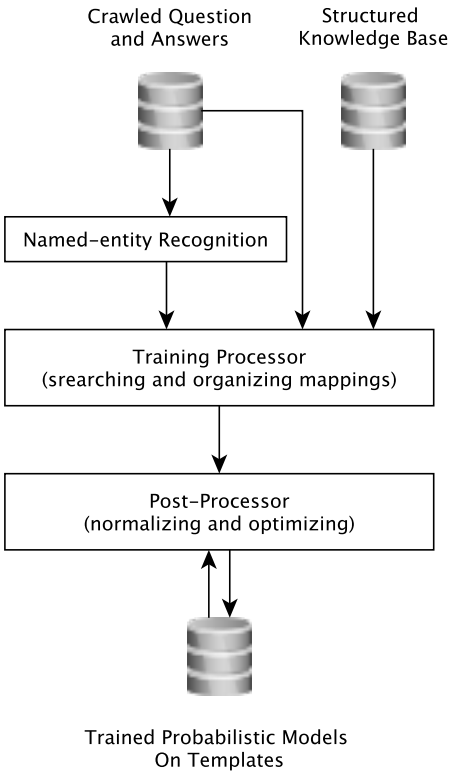

Figure 5.2: The Workflow of Unsupervised Training Process for mapping from natural language questions to structured queries

Figure 5.3 is a question answering workflow diagram illustrating a representative workflow in accordance with one aspect of the invention.

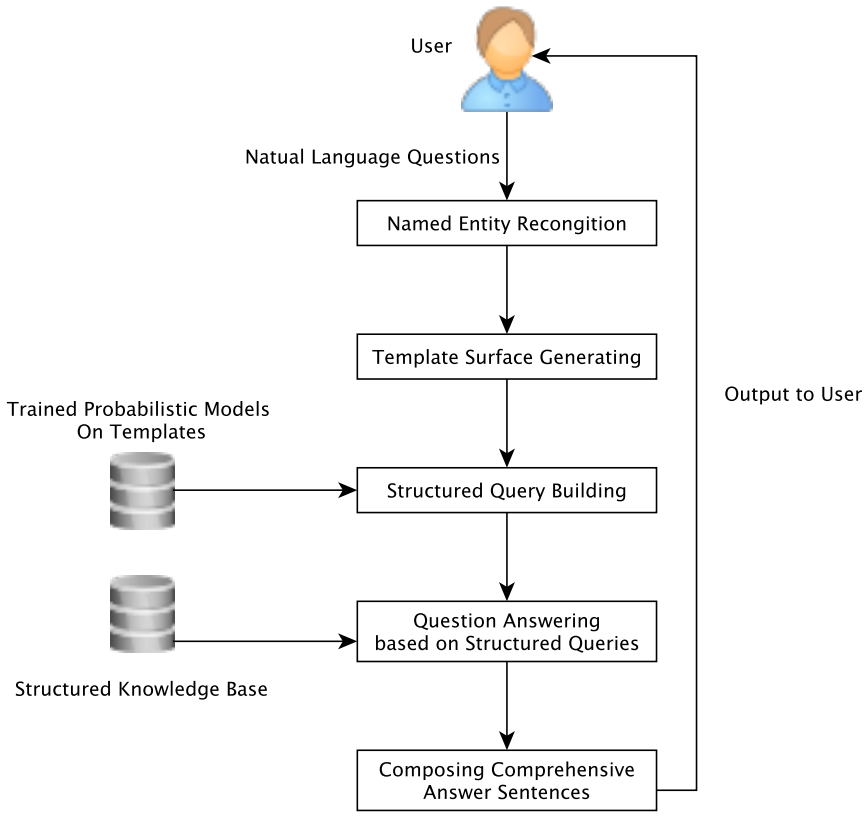

Figure 5.3: The Workflow of Ontology-based Question Answering

The present invention provides a computer network implemented system, a computer network implemented method, and a computer network architecture that is able to find concise answers for natural language questions with structured knowledge base, specifically mapping unstructured natural language (such as English) questions to structured queries and find its corresponding answer.

In one aspect of the invention, a computer implemented method is provided that learn somatic linkage between the natural language questions patterns and the structured queries they imply.

5.4 Name Dictionary and Triple Indexing

Most public structured knowledge sources are provided in plain triples form, such as N-Triples syntax [18] used by DBpedia. In the N-Triples syntax, concepts are expressed as URI (Uniform Resource Identifier) references. And each literal is in form of a string that optinally appended with a URI that indicates its data type. The figure 5.4 lists two example triples in DBpedia knowledge base.

```
Subject: <http://dbpedia.org/resource/Canada>
Predicate: <http://dbpedia.org/ontology/capital>
Object: <http://dbpedia.org/resource/Ottawa>

Subject: <http://dbpedia.org/resource/Canada>
Predicate: <http://dbpedia.org/ontology/foundingDate>
Object: ``1931-12-11''^^<http://www.w3.org/2001/XMLSchema#date>
```

Figure 5.4: Example Triples in Knowledge Base

For both time and speed efficiencies, a natural technique [10] is replacing all URI references and literals by numerical ids using a *name dictionary*. This approach enable faster and simpler triple indexing and querying operations. More importantly, the name dictionary delivers space economy. For instance, a path in knowledge graph can be straightforwardly stored in a sequence of id numbers. Usually a name dictionary introduces additional cost when translating queries in URIs and literals into its id. However, as the input of this system is natural language questions instead of structured queries, the Named-entity Recognition module (see Section 5.5) is able to map from surface texts to dictionary id directly, which eliminate the extra cost of using name dictionary.

As introduced in section 5.1, knowledge base are required to respond to basic pattern matching queries on triples as Basic Graph Patterns (BGP). Although most of the prior Ontology-based QA systems built on top of existing triplestore or relational database engines, this project optimizes triple searching procedure for QA tasks with a new triple storage and implementation from strach.

To enable efficient matching all triples by given BGP, indexes are need to be built from triples. There are six possible permutations of subject (S), predicate (P), and object (O),

which are SPO, PSO, OPS, POS, SOP, OSP. Because the BGPs in this application always contain two known elements and one unknown variable and query processor can adjust the order of known elements, only three indexes SPO, OPS, SOP are necessary and used. For example, the index SPO implements a mapping function: $I_{spo}(s,p) = \{o|(s,p,o) \in K\}$, where K is all triples in the knowledge base.

Hash table is chosen to be the underlying triple indexing data structure. Specifically, open addressing main memory hash tables are used with two known ids as key and data range as value. As shown in the figure 5.5, a supplementary array stores the value list of each key mapped. Based on a observation and assumption that each BGP query is

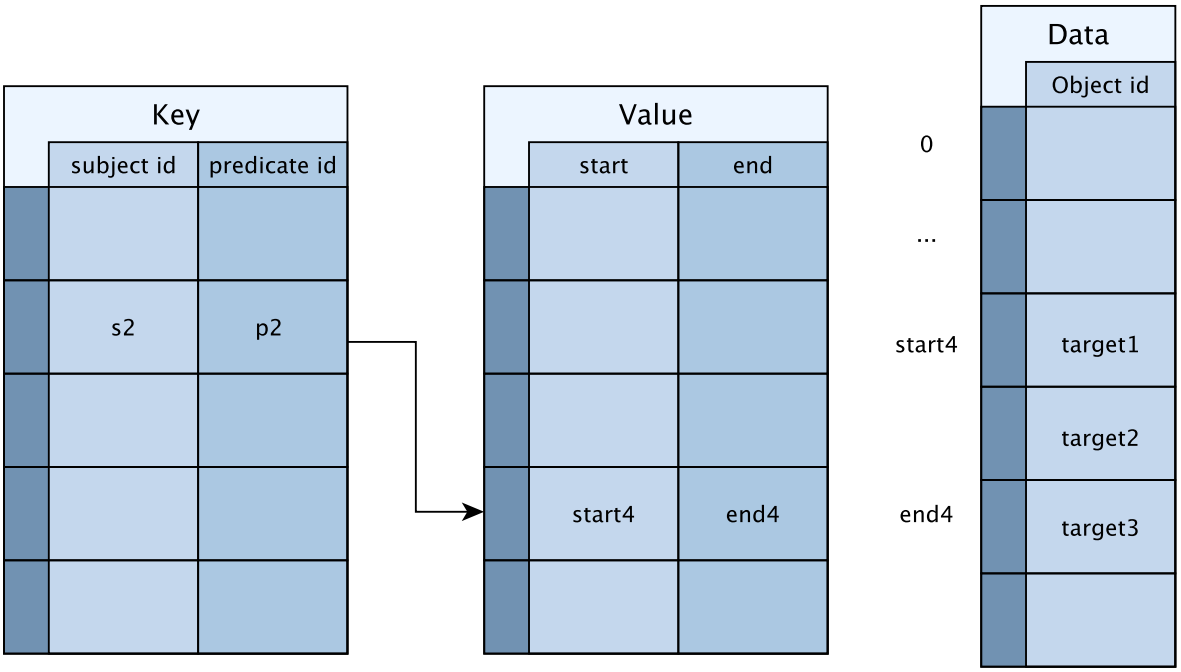

Figure 5.5: An Illustration of SPO Triple Index

independent, it is difficult to cache the index to memory well if indices are stored on disk. Because keys and values are all numeric ids, the main memory can more easily afford the hash table index even with the extra space consumed by empty slots, which avoids using

disk-friendly but slower looking up data structures like B^+-trees.

5.5 Named-entity Recognition based on Anchor Texts

The function of Named-entity Recognition (NER) module is to spot interesting text segments and bridge them to their probable semantic meanings in form of knowledge base entities. This NER module is frequently used by other modules such as identifying question topic in question answering module.

To support this open-domain large scale QA system, the NER is generated based on anchor text resources to meet needs of mapping miscellaneous textual expressions to knowledge entities. In many public available knowledge base such as wikipedia, each knowledge entity (topic) has its own public available URL. Therefore, webpages such as news, blogs, and wikepedia itself often add hyperlinks that point to related wikipedia topics within the passages. As the example shown in figure 5.6, a HTML web page might contain a hyperlink `<a href="http://en.wikipedia.org/wiki/Tracy_McGrady>T-Mac</a>` that link plain text "T-Mac" to the wikipedia page of "Tracy McGrady". In Wikitext format,

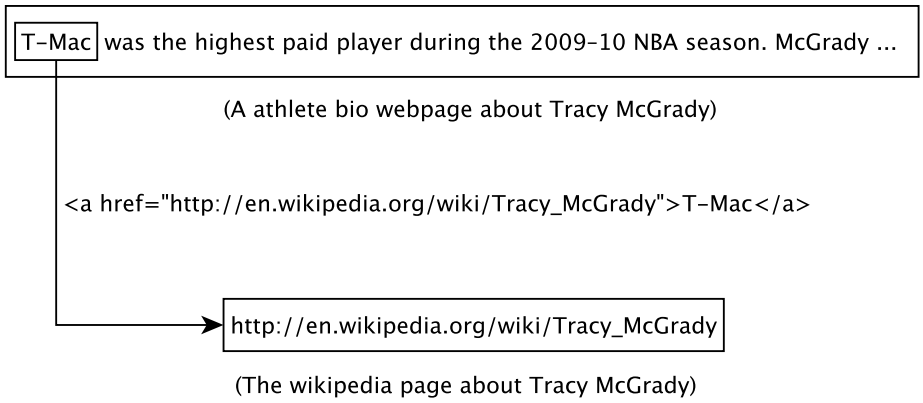

Figure 5.6: An example of anchor text link structure.

an equivalent link is formed as `[[Tracy McGrady|T-Mac]]`. In the above example, the

wikipedia topic page "Tracy McGrady" uniquely identify an entity in DBpedia knowledge base. Therefore, those surface texts (e.g. "T-Mac") in hyperlinks suggest textual labels or textual short expressions for an knowledge entity.

These are several advantageous characteristics of using those anchor surfaces as textual labels for entities. First, various surfaces can be extracted for same entity, which covers almost all kinds of possible labeling descriptions for each entity. Second, given a surface, the statistical distribution of pointed entities can act as the probability of each entity on this surface and can help disambiguate the surface. Third, for each entity, the total number of the reference links indicates the popularity of this entity.

A dictionary-based Named Entity Recognition (NER) is implemented with a prefix tree of words. The NER dictionary entries are extracted from links of articles as described above. Given input words sequence $(w_1, \ldots, w_n)$, the NER spots a set of surface texts

$$S = \{t_k = (w_i, \ldots, w_j) | 1 \leq i \leq j \leq n, 0 \leq k \leq n^2\} \tag{5.1}$$

that match a entry in surface dictionary. Since different entities can be referenced from same surface text, the NER produces a list of entities $E(t_k) = (e_1, \ldots, e_m)$ that possibly associating with each surface text t_k. Further, entities in the list are ranked based on the total count of each entity being referenced which likely imply its popularity.

In addition, the NER module also detects date, time, string literals, and other numbers such as monetary values. For example, numbers such as "3.1415 millions" will be extracted and approximately transformed to form as 3.14×10^6. The normalized number is stored and searched in the SOP index discussed in section 5.4, which enables approximate number matching between free text and knowledge base.

5.6 Templates Training

After applying NER on the unstructured question text, another critical challenge of successfully translation from natural language question to structured query is finding and selecting the proper relationship (i.e. predicate in RDF triple) to construct BGP. To overcome this challenge, this approach automatically learns this mapping rules, patterns, and statistics from natural language to proper relationship paths from question-answer pairs and knowledge bases.

The procedure of template training can be briefly described as following steps:

1. Given a pair of natural language question and its answer, the template training module applies NER to both question and answer. It collects spotted entity surfaces S_q from question and S_a from answer as described in section 5.5.

2. For every pair of surface texts (t_q, t_a) where $t_q \in S_q$ and $t_a \in S_a$.

   (a) The original question is normalized and grouped with spotted text t_q, which called template surface.

   (b) For every pair of entities (e_q, e_a) where $e_q \in E(t_q)$ and $e_a \in E(t_a)$.

      i. The training program exams whether there exists any short path between e_q and e_a in the graph of knowledge base.

      ii. If one or more short paths between e_q and e_a are discovered, those predicate paths and e_q will be recorded and attached to the template surface.

   (c) In addition, if more than one disjoint spotted texts in the question are found path to same entity in the answer, a template with multiple slots will be recorded.

   (d) However, if no short path is detected from all pairs of entity between $E(t_q)$ and $E(t_a)$, a special "no path found" path will be added to the template surface.

3. Repeat above steps over a collection of question and answer pairs. After finishing training all questions, for every template, a distribution over potential predicates paths is observed.

To illustrate, if NER extracts an entity textual surface $t_q = (w_i, \ldots, w_j)$ associating with entities $E(t_q)$ for the question and entities $E(t_a)$ for the answer, then the training program tries all queries with triple pattern (e_q, P, e_a) where $e_q \in E(t_q)$ and $e_a \in E(t_a)$. If predicate P fits one of above queries, the training program records this P appearance associating with a single slot template surface $T = (w_1, \ldots, w_{i-1}, M, w_{j+1}, \ldots, w_n)$, where M is a slot marker indicates the type of e_q. In other words, path P is added to path set $PS(T)$ for T and it counts a unit credit in counter $C_{P,T}$. By going through a large collection of pairs of questions and answers, the training program obtains a model containing predicate (relation) path distributions on each question template surface. The probability of predicate path r given

template T is the predicate path frequency:

$$p(r|T) = \mathrm{pf}_{\mathrm{p,T}} = \frac{C_{r,T}}{\sum_{k \in PS(T)} C_{k,T}} \tag{5.2}$$

For example, given template T_1 ="who is the president of [ORGANIZATION]?", the trained model can return a list of possible predicates with different probabilities such as $p(leaderName|T_1) = 0.5$ and $p(keyPerson|T_1) = 0.1$.

The unsupervised learning nature of above procedure and the noisy training data possibly introduce predicate paths that not semantically bridging a template to its answers. For example, when training with question "who is the mayor of Chicago" and its answer text "The current mayor of Chicago is Rahm Emanuel.", the predicate paths between "Chicago" and "Rahm Emanuel" are both $leaderName$ and $birthPlace^{-1}$. Even though many city mayors were indeed born at the city too, the $birthPlace^{-1}$ is not one of the desired predicate paths of the training. To overcome this problem, a relatively large collection of training questions and answers is necessary to provide statistical significance for predicate path distributions for templates. Specifically, the training data also contains questions about "who is mayor of Toronto" and "who is mayor of New York" that only related to $leaderName$ not $birthPlace^{-1}$, which makes predicate $leaderName$ statistically dominates for templates "who is mayor of [CITY]?".

In fact, the predicate paths observed during the training process have different credibilities. Several techniques are used for the above challenge by giving proper weight of each trained predicate path mapping. For instance, if c paths are detected between a pair of entities (e_q, e_a), the weight of each path in template path distribution is equally divided by c. The purpose of this operation is to deal with the semantically duplication in knowledge bases. Given training question "Who is Bill Gates' father?" and answer "William Gates", there exists two semantically equivalent predicate paths *parent* and $child^{-1}$ from "Bill Gates" to "William Gates". This normalization operation makes sure duplicated predicates won't gain extra credits, as contrasted with ordinary predicates which only appear once in one direction.

In addition, the likelihood of an entity for a surface text also affects the weights of a detected predicate path. Ideally, the probability of correct predicate path detection is proportional to the entity likelihood. However, in practice, the NER is not in a position to

disambiguate the surface text and produces accurate probability of candidate entities itself, which actually largely depends on its context. As a result, the weight of a predicate path is proportion of entities popularities in logarithmic scale. Therefore the weight function for a observed path with total c paths is:

$$w_{e_q,e_a} = \frac{1}{c} \times Q_{e_q,t_q} \times Q_{e_a,t_a} \tag{5.3}$$

with

$$Q_{e,t} = \frac{\log P_e}{\sum_{k \in E(t)} \log P_k} \tag{5.4}$$

where P_e is the popularity of an entity defined by NER module. In this way, for all pairs of entities $\mathrm{PE}_{p,T} = \{(e_q, e_a)\}$ that discovered predicate p for template T, the counter $C_{p,T}$ is added up with weight function 5.3 :

$$C_{r,T} = \sum_{(e_q,e_a) \in \mathrm{PE}_{p,T}} w_{e_q,e_a} \tag{5.5}$$

5.7 Normalization

The post-processing module employs multiple kinds of normalization and optimization on the trained model results.

First of all, standard morphological normalization is applied on the template surfaces. As a result, template surfaces with morphological variations of words are merged together. Distributions associated with those template surfaces are also combined. For example, all verbs will be normalized to simple present tense and in the plural form. Therefore, "Who marries PERSON" and "Who married PERSON" are normalized and merged to Who marry PERSON.

Secondly, templates surfaces within near tf-idf (term frequency-inverse document frequency) distance are merged together. The tf-idf is a well-tested statistical measure used to evaluate how important a word is to a document in corpus. Computing tf-idf in the collection of templates surfaces can provide empirical observations on each word's weights in for reflecting the topic of the template surface. The similarity between words sequences is defined as the cosine similarity of their if-idf vectors. Further, the distance between

word sequences is 1 minus the similarity value. Thus two templates are combined with their distribution if they satisfy all following conditions: 1) they share the same template slot marker (entity type); 2) their relation path (predicate) distribution overlaps in certain degree; 3) the tf-idf distance between them is near (e.g. less than 0.3). In short, this step merges the predicate distributions of those templates sharing major keywords and semantic mappings, which improve and unified the ability of distributions prediction.

One obvious observation at QA training pairs is that training answers contain noisy entities. Those noisy entities such as a country name can lead to unwanted predicate paths like *withinCountry*. In order to alleviate the effect of above problem, a predicate path which appears too often in templates should have lower weight than others. Similar with the idea of inverse document frequency, all predicates are applied with a weight function:

$$\text{itf}_{p,C} = \log \frac{|C|}{1 + |\{t \in C : p \in t\}|}, \tag{5.6}$$

where $|C|$ is the total number of templates trained and $|\{t \in C : p \in t\}|$ is the total number of templates that contains predicate path p. Besides, the predicate path frequency $\text{pf}_{\text{p,T}}$ for predicate path p in a template T is scaled by maximum predicate frequency in T like "maximum tf normalization" summarized in [46]. The normalized predicate path frequency is defined as

$$\text{npf}_{p,T} = \alpha + (1 - \alpha)\frac{\text{pf}_{\text{p,T}}}{\text{pf}_{max}(T)}, \tag{5.7}$$

where $\text{pf}_{max}(T) = \max_{p' \in T} \text{pf}_{p',T}$ and the parameter $\alpha \in (0, 1)$. This normalization will augment the probabilities of a few high rank predicate paths that likely to be picked to answer question. Two weighing function combined to make the score of a predicate path p in a template $T \in C$ to be:

$$\text{P-Score}_{p,T,C} = \text{npf}_{p,T} \times \text{itf}_{p,C} \tag{5.8}$$

In section 5.8, Equation 5.8 will be used to rank and choose answer.

5.8 Question Answering

Given the templates model trained in section 5.6, the basic questions can be answered by matching templates. Similar with the training process, a input question is first scanned

by NER to generate an entity surface $t_q = (w_i, \ldots, w_j)$ and then form a template for the input question $(w_1, \ldots, w_{i-1}, m, w_{j+1}, \ldots, w_n)$ with slot marker m. If the normalized input template is matched with at least one of trained templates, predicates is retrieved from the trained predicate path distribution of the matched template. With the entites refereed by NER and predicates mapped from trained templates, a group of BGPs then is generated. Groups of the BGPs might contain one or more BGP conjunction and are used to construct a comprehensive structured query. For instance, a predicate path $(father, brother^{-1})$ and a entity "Tom" can produce a basic query as shown in figure 5.7. Each BGP in the query

```
Select ?x where
{
        <Tom>  <father>  ?a .
        ?x     <brother> ?a .
}
```

Figure 5.7: A Example Generated Basic Query

is submitted to the SPO or OPS index built in section 5.4 to retrieval matched entities. If any result entity are found for input the query, it is added to the solution set as a candidate answer entity. Moreover, when multiple entity surfaces detected in the input question, it can either lead to one template with several slots or multiple candidate templates. If two entities are recognized and their surface texts are disjointed, a template with multiple slots are formed in similar way to single slot templates.

The present question answering method and system is able to find solutions for complex questions containing nested grammar structures. The templates training process only discovers short relation paths (e.g. less than 3 steps) for training efficiency. To answer complex questions, multiple steps structured queries are typically needed. One important observation is that nested template surfaces are equivalent to definition question template surfaces. For example, a definition question template surface "who is the daughter of PERSON" and another template surface "where did PERSON graduate from" can be used together for answering questions like "where did the daughter of PERSON graduate from". Therefore, the question answering module applies Context-Free Grammar (CFG) on template surfaces recursively to generate multiple steps structured queries and answer complex questions.

Assume the NER found a surface text $t_q = (w_{i2}, \ldots, w_{j2})$ in a question with word sequence $(w_1, \ldots, w_{i1}, \ldots, w_{i2}, \ldots, w_{j2}, \ldots, w_{j1}, \ldots, w_n)$. If the basic query built from template $(w_{i1}, \ldots, w_{i2-1}, m, w_{j2+1}, \ldots, w_{j1})$ and an entity $e_q \in E(t_q)$ successfully matches any answer entity e_a, the original question will be rewritten and be transformed to new template $T_{new} = (w_1, \ldots, w_{i1-1}, m, w_{j1+1}, \ldots, w_n)$. The same substitute procedure will be applied on T_{new} and e_a to obtain an further intermediate answer. The above route repeats endlessly until the original question is fully interpreted or the any intermediate query cannot be resolved. If the whole question is successfully parsed with templates along with a series of answer entities, this answer entity sequence will be added to the solution set as a multiple step answer.

Various kinds of answer entities are matched from different structured queries. For the purpose of ranking the entities corresponding to the question, scores are computed to measure the similarity between queries to the question. To this end, those structured queries are scored based on the popularity of nesting templates, the popularity of source entities, and the probability of predicate path corresponding to the template discussed in score 5.8.

The popularity of each template $T \in C$ is measured by template frequency $\mathrm{Tf}_{p,T}$, which is the sum of all its predicate path credits:

$$\mathrm{Tf}_{p,T} = \sum_{p \in T} (\mathrm{pf}_{p,T} \times \mathrm{itf}_{p,C}) \tag{5.9}$$

Further, for a multiple step answer which uses multiple templates, the credibility of a series of steps is decided by the least credible step following the Liebig's Law. Both P-Score and Tf reflect one step's credibility. Therefore, the overall score of a sequence of templates and predicates path pair $\mathrm{TP} = ((p_1, T_1), \ldots, (p_n, T_n))$ is:

$$\text{TP-Score}_{\mathrm{TP},C} = \min_{(p,T) \in \mathrm{PT}} (\mathrm{Tf}_{p,T} \times \text{P-Score}_{p,T,C}) \tag{5.10}$$

At this point, we may view the Tf as the total credibility of a template and the P-Score is the proportion of credit occupied by the predicate path.

In addition, the source entity frequency and number of returned answer entities also affects the final score of a solution. The source entity frequency for surface text t is defined

as:

$$\mathrm{Ef}_{e_q,t} = \frac{P_{e_q}}{\sum_{k \in E(t)} P_k} \tag{5.11}$$

where P_{e_q} is the popularity of the entity e defined by NER module. On account of the global observations on predicate path and template, source entity frequency play greater role on the credibility of a solution in contrast to the training process with a logarithmic scaled P_e. When a predicate path leads to a solution with many answer entities, this implies the possibility that this predicate path might also be counted redundant times during training. Besides, too many answer entities meet the preferences of QA systems on concise answers. Towards this end, we assign to weight w_{N_a} to a solution as well,

$$w_{N_a} = \frac{1}{1 + \log N_a}, \tag{5.12}$$

where N_a is the total number of answer entities in a solution.

Thus, the final score for each solution is combined with function:

$$\text{S-Score}_{\mathrm{TP},C,e_q,t,N_a} = \mathrm{Ef}_{e_q,t} \times \text{TP-Score}_{\mathrm{TP},C} \times w_{N_a} \tag{5.13}$$

Thus, solutions are ranked based on above score function and the highest scored solution is returned.

Since this ontology-based QA system is a sub-system of a larger QA system which explores various data sources, it is vital to respond "I don't know." for unsure question. Hence, two thresholds are used to decide whether choosing current best answer. The count $\mathrm{Tf}_{p,T}$ has to be larger than threshold θ_{Tf}. This criteria guarantee the statistical significant of this template mapping. Also, the S-Score should be larger than the θ_S to ensure that current solution is not recklessly chosen because of missing data in the knowledge base.

When necessary, the question answering module can employ an optional corpus-based answer validation and refines the module. When potential answers are found in the above modules, each answer and question keywords are examined by co-occurrence in the corpus. If the answer and keywords appear in the same passage frequently, then this answer phrase is validated.

As mentioned in weight function 5.3 during training, if c predicate paths are discovered a pair of training entities, each predicate path only gain $\frac{1}{c}$ its original credit in corresponding

counter. Though this weighing avoid duplicate predicate paths over counted, each duplicate predicate path alone has less count to compete with other predicate paths. To deal with this problem, when different queries from the same source entity reached the same answer entities, these solutions are merged to a new solution, and sum of their S-Scores is assigned to this new solution.

Instead of outputting the answer phrase directly, answer sentences are composed based on the answer phrase and its corresponding structured query. The present system provides a hand-written answer sentence template for the popular relation path. It is also able to make answer sentence by learning the relationship description from a corpus by summarizing textual description on known entities' surfaces.

5.9 Experiments

In this chapter, a natural language QA system is designed and implemented for querying ontology knowledge sources, which trained question analysis from community QA archive. Experiments are carried out to study the impact of community data on the performance of question answering. This section introduces the experimental environments, parameters tuning results, and performance evaluation of Question Answering.

5.9.1 Experiment Setup

The materials used to train the template mappings consists of 10 million question and answer pairs crawled from Yahoo! Answers and WikiAnswers websites. For questions with multiple available answers, only the best answer selected by community is used in for the training.

The knowledge base is built on RDF triples in DBpedia 3.7[1] (September 2011), which were extracted from Wikipedia articles infobox templates [1]. In the data set of DBpedia 3.7 release (September 2011), there are 3.64 million entities under 740,000 Wikipedia categories. The English portion of DBpedia consisting of 385 million RDF triples is indexed in this project as knowledge source.

[1] http://blog.dbpedia.org/2011/09/11/dbpedia-37-released-including-15-localized-editions/

A dictionary-based NER is constructed based on internal link surfaces in Wikipedia articles. In addition, the labels, redirects, and disambiguating triples ships with DBpdia data set is added to NER as well. Above data results in a NER dictionary with 3,580,963 surface entries.

TREC-9 (2000) QA track questions set [68, 67] is used to evaluate the end-to-end QA performance. For this test sets, to measure its ability to correctly classify true negative, questions that did not have known answers in the reference knowledge base are kept intact. TREC-9 test set includes 693 questions extracted from the logs of Microsoft's Encarta system and Excite search engine.

Because the knowledge source of this QA system is provided by DBpedia instead of TREC corpus, many answers for TREC questions is not available in this common sense knowledge base. Besides, this ontology-based QA system is designed to work with other QA systems together. As a result, the system is required to respond No Answer (NA) rather than inaccurate or unsure answers. The evaluation on system precision is the metric precision@1 defined as:

$$precision@1 = \frac{\#correctly\ answered}{\#answered\ question\ (not\ No\ Answer)} \tag{5.14}$$

And, to estimate the system's overall answering coverage, the percentage of correct answers over all inputs is used:

$$\%correct_{all} = \frac{\#correctly\ answered}{\#total\ input\ questions} \tag{5.15}$$

Since only a part of the answers exist in the DBpedia knowledge base, there are

$$\%correct_{human} = \frac{\#human\ can\ answered}{\#total\ input\ questions} \tag{5.16}$$

percent of questions in the testing set that a human searcher is able to answer by browsing DBpedia knowledge only. Moreover, the $\%correct_{human}$ for TREC-9 test set with DBpedia as knowledge base is measured as $\frac{377}{693} \approx 54.4\%$.

Therefore, $\%correct_{rel}$ is used to measure the recall of answer retrieval from this limited knowledge source, which is the percentage of correctly answered questions over questions that human is able to answered:

$$\%correct_{rel} = \frac{\#correctly\ answered}{\#human\ can\ answered} = \frac{\%correct}{\%correct_{human}} \tag{5.17}$$

5.9.2 Evaluation

As discussed in section 5.8, there are three thresholds to determine whether returning current best answer or replying no answer. In general, those thresholds balance the two evaluation metrics $precision@1$ and $\%correct_{rel}$. Intuitively, when those thresholds become more conservative, the QA system will achieve higher precision but possibly answer less number of questions, and vice versa. To meet different system requirements, two thresholds can be respectively tuned to optimize certain objectives function such as F1-Score.

To explain these thresholds in more detail meaning, the S-score is the objective function to rank answers, which measures the overall probability of the current answer to the input question. In another words, S-score reflects how close is current structured query mapping to the original question, and the θ_S control the maximum distance allowed between the question and structured query. On the other hand, the thresholds θ_{Tf} ensures the trustworthiness of S-Score. Table 5.1 and 5.2 summarizes the those effects in terms $precision@1$ and $\%correct_{rel}$.

Table 5.1: The impact of threshold θ_{Tf} when θ_S is fixed to 0.05

θ_{Tf}	Precision@1	$\%Correct_{rel}$
0	0.8206	0.4854
10	0.8198	0.4828
20	0.8341	0.4801
30	0.8429	0.4695
40	0.8510	0.4695
50	0.8502	0.4668
60	0.8607	0.4589
100	0.8724	0.4536
200	0.8778	0.4191
400	0.8957	0.3873
800	0.8951	0.3395

Compare Table 5.1 and 5.2, we can find that the system performance is clearly more sensitive on θ_S threshold. Further, it shows relatively small number of Tf is able to provide

Table 5.2: The impact of threshold θ_S when θ_{Tf} is fixed to 30

θ_S	Precision@1	$\%Correct_{rel}$
0	0.7300	0.5093
0.005	0.7510	0.5040
0.01	0.7602	0.4960
0.02	0.7881	0.4934
0.03	0.8009	0.4801
0.04	0.8241	0.4721
0.05	0.8429	0.4695
0.06	0.8731	0.4562
0.07	0.8789	0.4430
0.08	0.8824	0.4377
0.09	0.8827	0.4191
0.1	0.8908	0.4111
0.15	0.9250	0.3926
0.2	0.9504	0.3554
0.3	0.9655	0.2971
0.5	0.9877	0.2122

good confidence on template correctness, which is to prove it not trained from noise data pairs.

As shown, this Ontology-based QA system has relatively high precision@1 that can arrives more than 90% with tuned parameters. This characters would let it plays safely with other QA systems using different knowledge source. Moreover, the $\%correct_{human}$ is considered as the upper bound of $\%correct$ and is measured as 54.4%. In this way, the system can retrieve majority of available answers and avoid false positive answers at the same time.

In term of speed, this system achieves high throughput of average 1213 questions per second, which make it scalable to face the challenge of real-world industry usage. This implementation of QA system is significantly faster than other DBpedia-based natural language query system by a few orders of magnitude. For example, the PowerAqua [45] requires an average 20.06 seconds to respond a question.

Chapter 6

Conclusions

6.1 Summary

This thesis demonstrated practical usage of community question answering archive data for automatic question answering task. In this project, the definition, objective, and characteristics of question answering from difficult sources are revised, further analyzed three classes of transformations by the concept that dividing question into topic and focus part.

This thesis introduced query extension methods by collecting keywords for similar question focus. Moreover, this extended queries are used as a feature for passage extraction task for question answering, which improves its recall rates.

A end-to-end question answering system is developed, which comprising: receiving a natural language question; recognizing one or more entity surfaces in the natural language question, and generating one or more corresponding template surface queries; constructing one or more structured queries based on trained models for queried templates and entities; finding and selecting one or more answer phrases using constructed structured queries on a knowledge base; and composing output for user based on one or more answer phrases.

During constructing one or more structured queries from trained models and entities comprises: creating a basic graph pattern (BGP) with trained predicate paths and entities; performing boolean algebra on multiple BGPs to better describe the question intent; and applying context-free grammar (CFG) on template surfaces to build nesting queries.

When selecting one or more answer phrases using constructed structured queries comprises: employing a ranking scheme considering following factors: popularity of the templates surface; popularity of entities; probability of predicate path corresponding to the template; and the depth of the nesting structured queries.

Further, a automatically learning mappings from a natural language question to a structured query, comprising: a named entity recognizer for detecting the entity surface in input text to link a text phrase to a knowledge concept; a training processor for searching and organizing mapping from a template surface to one or more predicate paths; a training post-processor for normalizing and optimizing trained mapping distributions for one or more templates.

The named entity recognizer is adapted to: extract entity surfaces from one or more webpage links to each knowledge concept; build a prefix-tree based dictionary on extracted entity surfaces to link knowledge concepts; and search inputted text with a surface dictionary to identity one or more potential knowledge concepts.

Post-processor for normalizing and optimizing is adapted to: perform morphological normalization on template surfaces; merge the predicate distributions of those templates sharing major semantic mappings and near in tf-idf distance.

6.2 Future Work

This project defines three class of Question-Query transform function. But, since equal and inference transformation functions have been widely researched and applied, this work only focus on correspondence transformation. In future, other two class of transformation should also included to further overcome the information gap between the question and answers.

The CRFs passage extraction model is an open models that can balance overlap features. This is a trivial improvement for the effectiveness of overall QA system that adding more features. However, obviously, large features could also slow the system performance.

Lastly, the post-process module employed multiple kinds of matrix analysis, transformation and decomposition on the trained model. Considering each template surface is a row and each relation path of the template is a column, the trained model can be regarded

as a matrix. Therefore, matrix analysis techniques can be applied to reduce noises and/or detect corelativeness in relationship path distributions.

References

[1] Sören Auer and Jens Lehmann. What Have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content. In *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, ESWC '07, pages 503–517, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] J Bian, Y Liu, Eugene Agichtein, and Hongyuan Zha. Finding the right facts in the crowd: factoid question answering over social media. *International World Wide Web Conference Com- mittee (IW3C2)*, pages 467–476, 2008.

[3] MW Bilotti, Paul Ogilvie, and Jamie Callan. Structured retrieval for question answering. *in information retrieval*, pages 351–358, 2007.

[4] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[5] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, September 2009.

[6] Stefan Büttcher, Clarke Charles, and Cormack Gordon. *Information Retrieval: Implementing and Evaluating Search Engines*. MIT Press, 2010.

[7] Xin Cao and Gao Cong. A generalized framework of exploring category information for question retrieval in community question answer archives. *World Wide Web Conference Com- mittee (IW3C2)*, (December 2005):201–210, 2010.

[8] Xi Chen, Shihong Chen, and Weiming Wang. Passage Extraction Using Subsequence-Based Query-Sensitive Maximum Cut. *2008 International Symposium on Knowledge Acquisition and Modeling*, pages 221–225, December 2008.

[9] F.Y.Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 26–33. Morgan Kaufmann Publishers Inc., 2000.

[10] Eugene Inseok Chong, Souripriya Das, George Eadon, and Jagannathan Srinivasan. An Efficient SQL-based RDF Querying Scheme. In *Proceedings of the 31st VLDB Conference*, pages 1216–1227, 2005.

[11] Philipp Cimiano, Peter Haase, and Jörg Heizmann. Porting natural language interfaces between domains. In *Proceedings of the 12th international conference on Intelligent user interfaces - IUI '07*, page 180, New York, New York, USA, January 2007. ACM Press.

[12] Philipp Cimiano, Peter Haase, Jörg Heizmann, Matthias Mantel, and Rudi Studer. Towards portable natural language interfaces to knowledge bases The case of the ORAKEL system. *Data & Knowledge Engineering*, 65(2):325–354, May 2008.

[13] GV Cormack, CLA Clarke, and CR Palmer. Fast automatic passage ranking (MultiText experiments for TREC-8). *Proceedings of the*, 2000.

[14] Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '05*, page 400, 2005.

[15] D. Damljanovic, M. Agatonovic, and H. Cunningham. FREyA: an Interactive Way of Querying Linked Data using Natural Language. In *Proceedings of 1st Workshop on Question Answering over Linked Data (QALD-1), Collocated with the 8th Extended Semantic Web Conference (ESWC 2011), Heraklion, Greece*, 2011.

[16] Leon Derczynski, Jun Wang, Robert Gaizauskas, and Mark A. Greenwood. A data driven approach to query expansion in question answering. *Proceeding IRQA '08*

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering, pages 34–41, August 2008.

[17] T. Dkaki, J. Mothe, and Quoc Dinh Truong. Passage Retrieval Using Graph Vertices Comparison. In *Signal-Image Technologies and Internet-Based System, 2007. SITIS '07. Third International IEEE Conference on*, pages 71–76, 2007.

[18] Jan Grant and Dave Beckett. N-Triples syntax in W3C Working Draft "RDF Test Cases".

[19] Mark Greenwood. Using Pertainyms to Improve Passage Retrieval for Questions Requesting Information About a Location. *In Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering*, pages 17 – 22, 2004.

[20] Erik Hatcher and Otis Gospodnetic. *Lucene in Action*. Manning Publications, December 2004.

[21] Gary G. Hendrix, Gary G Hendrix, Earl D Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. Developing a Natural Language to Complex Data. *ACM TRANSACTIONS ON DATABASE SYSTEMS*, 3:105 – 147, 1978.

[22] U Hermjakob and A Echihabi. Natural language based reformulation resource and web exploitation for question answering. *Proceedings of the Text Retrieval*, 2002.

[23] Ulf Hermjakob. Parsing and question classification for question answering. In *Proceedings of the workshop on ARABIC language processing status and prospects -*, volume 12, pages 1–6, Morristown, NJ, USA, July 2001. Association for Computational Linguistics.

[24] Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-yew Lin. Question Answering in Webclopedia. *IN PROCEEDINGS OF THE NINTH TEXT RETRIEVAL CONFERENCE (TREC-9*, pages 655 – 664, 2000.

[25] A Ittycheriah, M Franz, WJ Zhu, and A Ratnaparkhi. IBM's statistical question answering system. *NIST SPECIAL*, 2001.

[26] Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. Finding similar questions in large question and answer archives. *Proceedings of the 14th ACM international conference on Information and knowledge management - CIKM '05*, page 84, 2005.

[27] Jing Jiang and Chengxiang Zhai. Extraction of coherent relevant passages using hidden Markov models. *ACM Transactions on Information Systems*, 24(3):295–319, July 2006.

[28] Pawel Jurczyk and Eugene Agichtein. Discovering authorities in question answer communities by using link analysis. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management - CIKM '07*, page 919, New York, New York, USA, November 2007. ACM Press.

[29] Marcin Kaszkiel and Justin Zobel. Effective ranking with arbitrary passages. *Journal of the American Society for Information Science and Technology*, 52(4):344–364, 2001.

[30] Esther Kaufmann, Abraham Bernstein, and Renato Zumstein. Querix: A natural language interface to query ontologies based on clarification dialogs. In *5th International Semantic Web Conference (ISWC 2006)*, number November, pages 980–981. Citeseer, 2006.

[31] M.A. Khalid and Suzan Verberne. Passage retrieval for question answering using Sliding Windows. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, number August, pages 26–33, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

[32] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - ACL '03*, pages 423–430, 2003.

[33] John Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 282–289. Citeseer, 2001.

[34] GG Lee, S Lee, Hanmin Jung, BH Cho, Changki Lee, and BK. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. *NIST SPECIAL*, 2002.

[35] Baoli Li, Yangdong Liu, and Eugene Agichtein. CoCQA: co-training over questions and answers with an application to predicting question subjectivity orientation. *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, (October):937–946, 2008.

[36] H. Li, Y. Cao, J. Xu, Y. Hu, S. Li, and D. Meyerzon. A new approach to intranet search based on information extraction. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 460–468. ACM, 2005.

[37] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, volume 12, pages 1–7. Association for Computational Linguistics, December 2002.

[38] Marc Light, Gideon S. Mann, Ellen Riloff, and Eric Breck. Analyses for elucidating current question answering technology. *Natural Language Engineering*, 7(04), February 2002.

[39] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(04):343–360, February 2002.

[40] Jimmy Lin and Boris Katz. Question answering from the web using knowledge annotation and knowledge mining techniques. *Proceedings of the twelfth international conference on Information and knowledge management - CIKM '03*, (November):116, 2003.

[41] Jimmy Lin, Dennis Quan, Vineet Sinha, Karun Bakshi, David Huynh, Boris Katz, and D.R. Karger. What makes a good answer? The role of context in question answering. In *Human-computer interaction: INTERACT'03; IFIP TC13 International Conference on Human-Computer Interaction, 1st-5th September 2003, Zurich, Switzerland*, number September, page 25. Ios Pr Inc, 2003.

[42] Xiaoyong Liu and W. Bruce Croft. Passage retrieval based on language models. *Proceedings of the eleventh international conference on Information and knowledge management - CIKM '02*, page 375, 2002.

[43] Xiaoyong Liu, W. Bruce Croft, and Matthew Koll. Finding experts in community-based question-answering services. In *Proceedings of the 14th ACM international*

conference on Information and knowledge management - CIKM '05, page 315, New York, New York, USA, October 2005. ACM Press.

[44] V Lopez, V Uren, E Motta, and M Pasin. AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):72–105, June 2007.

[45] Vanessa Lopez and M Fernndez. Poweraqua: Supporting users in querying and exploring the semantic web content. *Semantic Web Journal*, 2011.

[46] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[47] Frank Manola and Eric Miller. RDF primer. *W3C recommendation*, 2004.

[48] Paul Martin, Douglas E. Appelt, Barbara J. Grosz, and Fernando Pereira. TEAM: an experimental transportable natural-language interface. pages 260–267, November 1986.

[49] Paul Mcnamee, Rion Snow, Patrick Schone, and James Mayfield. Learning Named-Entity Hyponyms for Question Answering. pages 799 – 804, 2008.

[50] M Melucci. Passage retrieval: A probabilistic technique. *Information Processing & Management*, 34(1):43–68, January 1998.

[51] George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, November 1995.

[52] E. Mittendorf and P. Schauble. Document and passage retrieval based on hidden Markov models. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–327. Springer-Verlag New York, Inc., 1994.

[53] Thomas S. Morton. Using coreference for question answering. *Proceeding CorefApp '99 Proceedings of the Workshop on Coreference and its Applications*, pages 85–89, June 1999.

[54] Matte Negri and Milen Kouylekov. Question Answering over Structured Data: an Entailment-Based Approach to Question Analysis. *International Conference RANLP 2009 - Borovets, Bulgaria*, pages 305–311, 2009.

[55] Marius A. Pasca and Sandra M. Harabagiu. High performance question/answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '01*, pages 366–374, New York, New York, USA, September 2001. ACM Press.

[56] Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. Modern natural language interfaces to databases: composing statistical parsing with semantic tractability. In *Proceedings of the 20th international conference on Computational Linguistics - COLING '04*, pages 141–es, Morristown, NJ, USA, August 2004. Association for Computational Linguistics.

[57] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces - IUI '03*, page 149, New York, New York, USA, January 2003. ACM Press.

[58] John Prager, Eric Brown, Anni Coden, and Dragomir Radev. Question-answering by predictive annotation. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '00*, pages 184–191, New York, New York, USA, July 2000. ACM Press.

[59] Eric Prudhommeaux and Andy Seaborne. SPARQL Query Language for RDF, 2008.

[60] Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. Statistical machine translation for query expansion in answer retrieval. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 464, 2007.

[61] Gerard Salton, J Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–58. ACM, 1993.

[62] Stuart C. Shapiro. Encyclopedia of Artificial Intelligence. January 1992.

[63] Dan Shen and Mirella Lapata. Using semantic roles to improve question answering. *Proceedings of EMNLP-CoNLL*, (June):12–21, 2007.

[64] Xiance Si and EY Chang. Confucius and its intelligent disciples: integrating social with search. *Proceedings of the VLDB*, 3(2):1505–1516, 2010.

[65] Charles Sutton and A McCallum. An introduction to conditional random fields for relational learning. in introduction to statistical relational learnin. *Graphical Models*, (x), 2006.

[66] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. Quantitative evaluation of passage retrieval algorithms for question answering. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval - SIGIR '03*, (July):41, 2003.

[67] E Voorhees. Overview of the TREC 2001 question answering track. *NIST SPECIAL PUBLICATION SP*, 2002.

[68] Ellen M. Voorhees and Dawn M. Tice. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '00*, pages 200–207, New York, New York, USA, July 2000. ACM Press.

[69] Chong Wang, Miao Xiong, Qi Zhou, and Yong Yu. Panto: A portable natural language interface to ontologies. *The Semantic Web: Research and Applications*, pages 473–487, 2007.

[70] Kai Wang and Zhaoyan Ming. A syntactic tree matching approach to finding similar questions in community-based qa services. *Proceedings of the 32nd international ACM*, page 187, 2009.

[71] W. A. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, October 1970.

[72] William A. Woods, Robert M. Kaplan, and Bonnie Lynn Nash-Webber. The Lunar Sciences Natural Language Information System: Final Report. *Technical Report BBN Report, Bold Beranek and Newman Inc., Cambridge, Massachusetts*, 2378, 1972.

[73] Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. Retrieval models for question and answer archives. *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '08*, page 475, 2008.

[74] John M. Zelle and Raymond Mooney. Learning Semantic Grammars with Constructive Inductive Logic Programming. May 1993.

[75] John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. pages 1050–1055, August 1996.