

Interpretation of Natural Language using Data Mining, NLP and Machine Learning Techniques

A THESIS

submitted by

ANISHA KUMARI B120535CS

DRISHYA PRAVEEN C P B120086CS

INDU SREE B120668CS

in partial fulfilment for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

under the guidance of

Dr G. Gopakumar



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY CALICUT
NIT CAMPUS P.O, CALICUT
KERALA, INDIA 673601

May 9, 2016

Abstract

Our project aims at building a prototype capable of answering questions posed in natural language, motivated by IBM Tool called WATSON. Watson has a knowledge base of structured and unstructured data. When posed with a question, it searches for the possible keywords in the query with what is present in its knowledge base. Normal keyword based search engines retrieves results which are optimistic yet they lack in their ability to interpret user question.

Contents

1	Introduction	1
1.1	Problem Statement	1
1.2	Literature Survey	2
1.2.1	WatsonPaths: Scenario-based Question Answering and Inference over Un-structured Information	2
1.2.2	Ontology Based Information Retrieval System for Academic Library . . .	2
1.2.3	Tools and Methods for Building Watson	2
1.2.4	Natural language question answering: the view from here	2
2	Design	4
3	Implementation	7
4	Results	12
5	Conclusions	17

List of Figures

3.1	Ontology	8
3.2	Triplet Extraction Algorithm	9
3.3	Tone of a Passage	10
3.4	Mood of a Passage	11
4.1	Input question	12
4.2	Tokens	12
4.3	Tokens with POS tags	12
4.4	Chunk Grammar example	13
4.5	Parse tree	13
4.6	Triple (Subject-Predicate-Object)	13
4.7	SPARQL query	14
4.8	Ontology	14
4.9	Tone Input 1	14
4.10	Tone Output 1	15
4.11	Tone Input 2	15
4.12	Tone Output 2	15
4.13	Mood Input 1	16
4.14	Mood Output 1	16
4.15	Mood Input 2	16
4.16	Mood Output 2	16

Chapter 1

Introduction

A keyword based search engine presents to user enormous amount of data on being asked a question from which he or she cannot figure out the essential and most important information. Question answering system is a form of information retrieval system which aims to deliver the exact answer to the user question rather than whole document. To answer this user need semantic based reformulation techniques that can be used to retrieve the accurate answer from the enormous number of documents retrieved from the search engine. Watson is an artificial intelligent system that answers questions posed in natural language. It was developed by IBM specifically to answer questions in a quiz show called Jeopardy and it applies advanced natural language processing, information retrieval and machine learning to the field of question answering. It has millions of structured and unstructured data in its storage. It parses keywords while searching for related terms in a clue. It makes use of Map-Reduce Algorithm.

It has Applications beyond Jeopardy too, such as :

- Watson can be used to fast-track life-saving research, create connections in cold cases all in a matter of seconds, not in weeks or months.
- Watson for Clinical Trial Matching : Uncover new promising approaches often unavailable outside of the clinical trial setting.
- Watson Curator : Optimize business information to increase user confidence in Watson responses.
- Watson for Oncology : Assistance for oncologists to make more informed treatment decisions.

1.1 Problem Statement

The problem is to develop a miniature prototype of IBM tool Watson using Machine learning Techniques, Language Analysis Algorithms with the help of Apache Jena framework.

1.2 Literature Survey

1.2.1 WatsonPaths: Scenario-based Question Answering and Inference over Unstructured Information

[1] WatsonPaths can answer scenario-based questions (The goal of scenario analysis is to identify information in the natural language narrative of the problem scenario that is potentially relevant to solving the problem), for example, medical questions that present a patient summary and ask for the most likely diagnosis or most appropriate treatment.

Watson paths build on the IBM Watson question answering system that takes natural language questions as input and produces precise answers along with accurate confidences as output. It breaks down the input scenario into individual pieces of information, asks relevant sub questions of Watson to conclude new information and represents these results in a graphical model i.e. assertion graph. Probabilistic inference is performed over the graph to conclude the answer and is repeated until a stopping condition is met.

WatsonPaths takes a two-pronged approach to medical problem solving - First, by expanding the graph forward from the scenario in an attempt to make a diagnosis. Then linking high condence diagnosis with the hypothesis.

1.2.2 Ontology Based Information Retrieval System for Academic Library

[2] The paper describes in detail the development of a search engine that interprets the meaning of user's query instead of a keyword based search produces list of answers instead of specific answer. In this paper, an ontology based semantic information retrieval system is proposed utilizing Jena semantic web framework and Protege. A user enters an input query which is parsed by Stanford Parser followed by application of triplet extraction algorithm. Then SPARQL query is formed and it is fired on the knowledge base (ontology) which finds appropriate RDF Triples in knowledge base and retrieve the relevant information using Jena Semantic Web Framework.

1.2.3 Tools and Methods for Building Watson

[3] This paper discusses the methodology followed by research team during the development of Watson. The paper describes in detail the software environment, software integration and testing protocol, the DeepQA computing environment built to support huge volumes of high-throughput experiments and the tools they built for system performance measurement and error analysis.

1.2.4 Natural language question answering: the view from here

[4] It provides an overview of question answering as a research topic in terms of Applications, Users, Question types, Answer types, Evaluation, Presentation. There are also different methodologies for constructing an answer: through extraction, cutting and pasting snippets from the original documents containing the answer or via generation.

This article discusses about 3 major activities :

- Information retrieval(IR)
- Information extraction(IE)
- Question Answering

IR techniques have been extended to return not just relevant documents, but relevant passages within documents. The IR community has developed an extremely thorough methodology for evaluation which resulted in development of question answering evaluation. IE may be viewed as a limited form of question answering in which the questions (templates) are static and the data from which the questions are to be answered are an arbitrarily large dynamic collection of texts.

It explains Generic Architecture for a Question Answering System and the steps involved in it :

- Question Analysis
- Document Collection Preprocessing
- Candidate Document Selection
- Candidate Document Analysis
- Answer Extraction
- Response Generation

Chapter 2

Design

Domain:

A short story : "The Tiger King"

Database:

The Story is in the form of unstructured text. Apart from the unstructured text database, a table is maintained containing all Generic and Specific questions. Under the Generic questions, we will be maintaining general questions regarding the details of the story and its book like - name, author, name of the publisher etc and its answers. In Specific questions, we will be writing all possible questions and summary corresponding to each paragraph of the book.

Ontology:

Ontological database for storage of book information using Protege tool is created and this database is used to retrieve related answer from the domain specific ontology.

User interface is created in which user enters input question in natural language. When a question is posed, it is first searched in both Generic and Specific Questions table and if a match is found, then a corresponding answer is returned to the user else proceed with the following steps.

SYNTAX ANALYSIS

Tokenization

Input: Question

Process:

Question is subdivided into tokens using tokeniser which includes lemmatisation using NLTK library by making use of Snowball stemmer. It is written in Python language.

Output: Tokens

POS Tagging

Input: Tokens

Process:

Each token is associated with its part-of-speech tags using NLTK. Some of the POS tags are NN - Noun Singular VB - Verb Base Form NNS - Noun Plural DT -Determiner

Output: Tokens with their corresponding POS tags

Chunking

Input: Tokens with POS tags,Chunk Grammar

Process:

Chunk Parser is formed using Chunk Grammar which segments and labels multi-token sequences. Chunk Parser is used to construct tree structure of question.

Output: Parse Tree

SEMANTIC ANALYSIS

WordNet is a lexical database for the English language. WordNet can help a question answering system to identify synonyms. The synonym information can be used to help match a question with an appropriate rule.

RDF Triple Extraction

Input: Parse Tree

Process :

Triplet Extraction Algorithm is used to extract Subject, Predicate, Object from the tree structure.

1. A sentence (S) is represented by the parser as a tree having three children: a noun phrase (NP), a verbal phrase (VP) and the full stop (.). The root of the tree will be S. Firstly we intend to find the subject of the sentence. In order to find it, we are going to search in the NP subtree. The subject will be found by performing breadth first search and selecting the first descendent of NP that is a noun.
2. Secondly, for determining the predicate of the sentence, a search will be performed in the VP subtree. The deepest verb descendent of the verb phrase will give the second element of the triplet.
3. Thirdly, we look for objects. These can be found in three different subtrees, all siblings of the VP subtree containing the predicate. The subtrees are: PP (prepositional phrase), NP and ADJP (adjective phrase). In NP and PP we search for the first noun, while in ADJP we find the first adjective.

Output: Triple

SPARQL Query Generation

Input: Triple

Process:

Before generating SPARQL query, WordNet or local dictionary can be used to identify synonyms for predicates. If matching predicate is not found in RDF database, Query is generated using Apache Jena framework.

Output: SPARQL query

Information Extraction

Input: SPARQL Query, Ontology

Process:

Jena provides SPARQL API to handle both SPARQL query and their update. Jena API enables the SPARQL query for mapping with RDF database then it is fired on RDF database and retrieves the relevant information performing semantic search into database.

Output: Answer

Chapter 3

Implementation

1. Our project aims at building a prototype capable of answering questions posed in natural language.
2. Ontological database for storage of book information using Protege tool has been created and this database is used to retrieve related answer from the domain specific ontology.
3. User interface has been created in which user enters input question in natural language. When a question is posed, it is first searched in both Generic and Specific Questions table and if a match is found, then a corresponding answer is returned to the user else proceed with Syntax and Semantic Analysis.
4. SYNTAX ANALYSIS
 - Query has been subdivided into tokens using tokeniser which includes lemmatisation using NLTK library by making use of Snowball stemmer.
 - Syntax Analysis phase has been done using Stanford Parser and NLTK. Operations like – Tokenization, Chunking, POS Tagging were performed on the input questions for getting the parse tree.
 - Chunk Parser has been used to construct tree structure of question. It is formed using Chunk Grammar which segments and labels multi-token sequences.
5. SEMANTIC ANALYSIS
 - Triplet Extraction Algorithm has been used to extract Subject, Predicate, Object from the tree structure.
 - RDF Triples have been generated using Triplet extraction Algorithm in the Semantic Analysis phase.
 - Jena Apache Framework has been used to provide SPARQL API to handle both SPARQL query and their update. It enables the SPARQL query for mapping with RDF database, then it is fired on RDF database and retrieves the relevant information performing semantic search into database.
6. DATABASE
 - A short story "The Tiger King" has been taken. All possible questions from each paragraph with their answers are made and a database is populated. After this a

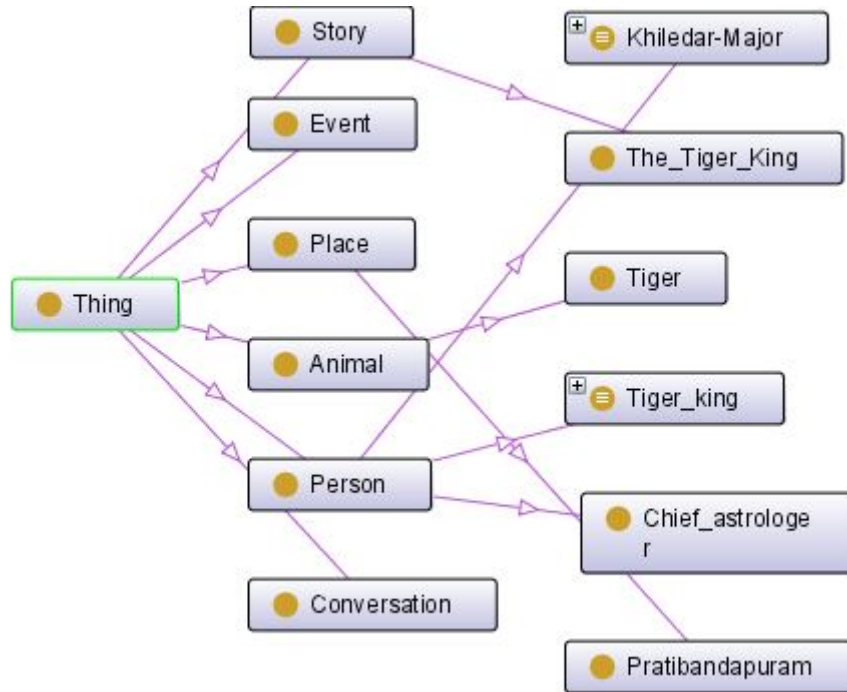


Figure 3.1: Ontology

User Interface is created (using HTML, PHP, CSS) where a user can get answers to the direct questions if it is already present in the database.

7. Algorithm to find the mood of a passage has been written and implemented using Bag of Words on random passages. Sample answers are like – Mood is Positive / Negative / Neutral. For implementing this algorithm, separate databases of large number of positive and negative words have been created. When a passage is given as input, the algorithm calculates the number of positive and negative words and displays the mood accordingly.
8. For finding Tone of a passage, a database has been created. It contains many passages with their respective tones which act as a Training set. 5 basic tones have been included in the database namely- Happy, Fear, Motivational, Technical and Aggressive. Using Naive bayes, probability of a random passage is found out and the result is displayed accordingly. The one having maximum probability is the answer.

Here are some of the Algorithms(pseudo codes) which we have implemented :

1. Triplet-Extraction Algorithm
2. Finding Tone of a Passage
3. Finding Mood of a Passage

Algorithm 1 TRIPLET EXTRACTION ALGORITHM

INPUT : Parse tree format of the Question

OUTPUT : Triplets

```
function NP_SUBTREE(sentence)
  returns the first subtree with NP as the root
```

```
function VP_SUBTREE(sentence)
    returns the first subtree with VP as the root
```

function PP_SUBTREE(sentence)
returns the first subtree with PP as the root

```

function EXTRACT_OBJECT(sentence)
    if the sentence contains (`` ``) //extracting dialogues
        object = all the following words till (" ")
    else
        if VPsubtree is not present
            object=deepest noun of PP subtree
        else
            Find NP, PP, ADJP, JJR subtrees from VP subtree
            for each value in the subtrees do
                if value = NP or PP
                    object= first noun in value
                else
                    object=first adjective in value
    return object

```

```
function EXTRACT_PREDICATE(sentence)
    if VPsubtree is absent
        predicate=deepest noun of NP subtree which is the sibling of PP subtree
    else
        predicate=deepest verb found in VP subtree which are of the form VB, VBD,
        VBG, VBN, VBP, etc
    return predicate
```

```
function EXTRACT_SUBJECT(sentence)
    if SBARQ subtree is present
        subject=nouns found in WP or WDT subtrees //extracting who, what, which
    else
        subject=first noun found in NP subtree which are of the form NNP, NNPS,
        PRP, NNS etc
    return subject
```

Figure 3.2: Triplet Extraction Algorithm

Algorithm 2 TONE OF A PASSAGE

Input : Passage
Training set : Files containing passages for each tone (tonefiles)
Selected tones : Happy, Motivation, Technology, Fear, Aggressive
Output : Tone of passage

function count_words (tonefile)
returns the number of words in the file

function count_aword(tonefile,word)
returns the number of occurrence of a particular word in the file

function voc(tonefile)
returns the vocabulary of the file

main function

for each tonefile fi **do** //counting number of words of each tone file
qi=count_words(fi)

for each tonefile fi **do** //counting vocabulary of each tone
vi=voc(fi)

for each tone file fi **do**
 for each word wj in fi **do**
 p=count_aword(wj,fi)
 sum+=log2((p+1)/(qi+vi)); // Naive Bayes formula using Add one smoothing

Ci=log2 (1.0/5.0) + sum // Probability of each tone

M=Max(Cis')

Return Tone corresponding to M //tone having maximum Ci (probability)

For passages belonging to more than one tone, a threshold value is set.
Tones > threshold value → Selected
Tones < threshold value → Rejected

Figure 3.3: Tone of a Passage

Algorithm 3 MOOD OF A PASSAGE

Input : Passage
Training set : Files containing + and - words (positive and negative files)
Output : Mood of passage

main function

```
for each word  $w_i$  of the passage do
    count_pos += count( $w_i$ ) in positive file
    count_neg += count( $w_i$ ) in negative file

if( count_pos > count_neg )
    mood = positive
else if ( count_pos < count_neg )
    mood = negative
else
    mood = neutral
```

Figure 3.4: Mood of a Passage

Chapter 4

Results

SYNTAX ANALYSIS

Tokenization

Input: Question

```
Who is the king of Pratibandapuram ?
```

Figure 4.1: Input question

Output: Tokens

```
['Who', 'is', 'the', 'king', 'of', 'Pratibandapuram', '?']
```

Figure 4.2: Tokens

POS Tagging

Input: Tokens

Output: Tokens with their corresponding POS tags

```
[('Who', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('king', 'NN'), ('of', 'IN'), ('Pratibandapuram', 'NNP'), ('?', '.')]
```

Figure 4.3: Tokens with POS tags

Chunking

Input: Tokens with POS tags

Chunk Grammar

NP: {<DT>?<JJ>*<NN>}

Figure 4.4: Chunk Grammar example

Output: Parse Tree

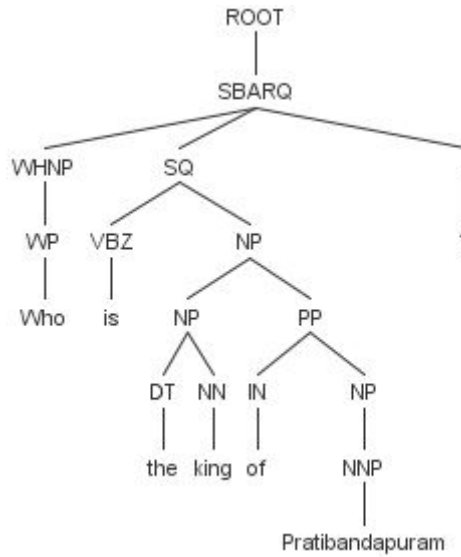


Figure 4.5: Parse tree

SEMANTIC ANALYSIS

RDF Triple Extraction

Input: Parse Tree

Output: Triple

```
SUBJECT:who
PREDICATE:king
OBJECT:pratibandapuram
```

Figure 4.6: Triple (Subject-Predicate-Object)

SPARQL Query Generation

Input: Triple

Output: SPARQL query

```

SELECT ?subject
WHERE { ?subject
  <http://www.semanticweb.org/dell/ontologies/2016/3/tiger_king#king_of>
  |<http://www.semanticweb.org/dell/ontologies/2016/3/tiger_king#Pratibandapuram>
}

```

Figure 4.7: SPARQL query

Information Extraction

Input: SPARQL Query

Ontology

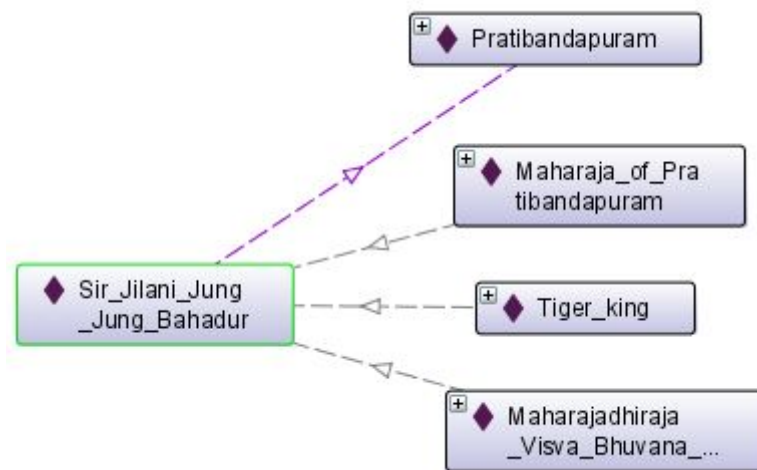


Figure 4.8: Ontology

Output: Sir Jilani Jung Jung Bahadur

IDENTIFICATION OF TONE

Input: Passage

"It's watchin' us" Ralphie whispered.
 "Listen, I'm not gonna-"
 "No, Danny. Really. Can't you feel it?"
 Danny stopped. And in the way of children, he did feel something
 and knew they were no longer alone. A great hush had fallen over
 the woods; but it was a malefic hush. Shadows, urged by the
 wind, twisted languorously around them.

Figure 4.9: Tone Input 1

Output:

```
Happy :-603.726422
Technical :-634.013722
Fear :-547.080896
Motivating :-608.141651
Aggressive :-611.409799
Threshold: -600
***FEAR***
```

Figure 4.10: Tone Output 1

Input: Passage

Each step you take toward a bigger goal might not seem like much. It may seem like you're not really doing much at all. This will be especially true of any outside observers. Others might think you're not getting anywhere, that you're not getting anything done. That's why you have to have a lot of confidence in where you're going. You need to be clear about where you want to end up so that you have the conviction that the small steps you are taking will eventually get you to where you want to be, and you can squash any naysayers.

Figure 4.11: Tone Input 2

Output:

```
Happy :-962.084462
Technical :-1012.975948
Fear :-960.751470
Motivating :-929.790504
Aggressive :-977.591932
Threshold: -968
***MOTIVATING***
```

Figure 4.12: Tone Output 2

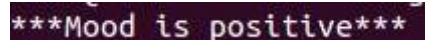
IDENTIFICATION OF MOOD

Input: Passage

But I feel peaceful. Your success in the ring this morning was, to a small degree, my success. Your future is assured. You will live, secure and safe, Wilbur. Nothing can harm you now. These autumn days will shorten and grow cold. The leaves will shake loose from the trees and fall. Christmas will come, and the snows of winter. You will live to enjoy the beauty of the frozen world, for you mean a great deal to Zuckerman and he will not harm you, ever. Winter will pass, the days will lengthen, the ice will melt in the pasture pond. The song sparrow will return and sing, the frogs will awake, the warm wind will blow again. All these sights and sounds and smells will be yours to enjoy, Wilbur—this lovely world, these precious days..."

Figure 4.13: Mood Input 1

Output:



```
***Mood is positive***
```

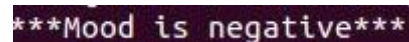
Figure 4.14: Mood Output 1

Input: Passage

And the trees all died. They were orange trees. I don't know why they died, they just died. Something wrong with the soil possibly or maybe the stuff we got from the nursery wasn't the best. We complained about it. So we've got thirty kids there, each kid had his or her own little tree to plant and we've got these thirty dead trees. All these kids looking at these little brown sticks, it was depressing.

Figure 4.15: Mood Input 2

Output:



```
***Mood is negative***
```

Figure 4.16: Mood Output 2

Chapter 5

Conclusions

The aim of our project is to build a miniature prototype which is capable of answering questions posed in natural language, motivated by an IBM tool called WATSON. Watson has a knowledge base of structured and unstructured data. When posed with a question, it searches for the possible keywords in the query with what is present in its knowledge base. Normal keyword based search engines retrieves results which are optimistic yet they lack in their ability to interpret user question. In this era, a search done by interpreting its meaning is more advantageous than that of blind keyword search.

The project goes through two stages mainly Syntactic Analysis and Semantic Analysis. Syntactic phase deals with Tokenisation, POS tagging and Chunking, which is taken care by Stanford Parser. The output of this phase is a Parse Tree.

The Semantic Phase involves understanding the meaning of the query. It involves Triplet Extraction, SPARQL query generation followed by Information Retrieval. The Triplets (subject-predicate-object) are extracted using Triplet Extraction Algorithm which is fed into SPARQL generation phase and corresponding SPARQL query is formed. This query is then fired upon ontological database which returns the required answer.

As a modification of the project, two distinct features have been added - a.Finding the Mood of a passage b.Finding the Tone of a passage. For mood, sample answers are like – Mood is Positive/Negative/Neutral. For tone, 5 basic tones have been included namely- Happy, Fear, Motivational, Technical and Aggressive. Probability of all the tones of a random passage is found out and the one having maximum probability is the answer.

The project can be enhanced and upgraded by adding a few more features like - Passage Summarization, Finding Moral and Scope, etc. It can also be remodeled by implementing in other languages(other than English). These have not been covered in the project currently because of the time constraint.

Bibliography

- [1] Adam Lally, Sugato Bachi, Michael A. Barborak, David W. Buchanan, Jennifer Chu-Carroll, David A. Ferrucci*, Michael R. Glass, Aditya Kalyanpur, Erik T. Mueller, J. William Murdock, Siddharth Patwardhan, John M. Prager, Christopher A. Welty. *WatsonPaths: Scenario-based Question Answering and Inference over Unstructured Information*, RC25489 (WAT1409-048) September 17, 2014.
- [2] Amol N. Jamgade and Shivkumar J. Karale, *Ontology Based Information Retrieval System for Academic Library* ,IEEE Sponsored 2nd International Conference on Innovations in Information, Embedded and Communication systems (ICIECS) 2015.
- [3] Eric Brown, Eddie Epstein, J. William Murdock, Tong-Haing Fin, *Tools and Methods for Building Watson* , RC25356 (WAT1302-021) February 15, 2013.
- [4] L.Hirschman, R.Gaizauskas, *Natural language question answering: the view from here* , 2001 Cambridge University Press.
- [5] Haiqing Hu, Peilin Jiang, Fuji Ren and Shingo Kuroiwa, *Web-based Question Answering System for Restricted Domain Based of Integrating Method Using Semantic Information* .